In [1]:
```
import pandas as pd
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
import csv
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import gensim
from gensim.models import Word2Vec
```

In [2]:
```
# Gerekli NLTK kaynaklarını indir (ilk çalıştırmada gerekli)
nltk.download('punkt', quiet=True)
nltk.download('stopwords', quiet=True)
```

Out[2]:
```
True
```

In [51]:
```
# [1] Kütüphaneleri içe aktarma
# Yukarıda yapıldı.
```

In [3]:
```
# [2] Veri setini yükleme ve ilk 500 karakteri gösterme
df = pd.read_csv('C:/Users/ademt/Desktop/tarim_problemleri_veriseti.csv', encoding='utf-8')
texts = df['sorun_metin'].tolist()
print("Verinin ilk 500 karakteri:")
print(''.join([str(text)[:500] for text in texts if isinstance(text, str)][:1]))
```

```
Verinin ilk 500 karakteri:
Bu yaz sezonunda şeftali tarlamızda yapraklarda kahverengileşme ve büzüşme. Sulama artırılmasına rağmen bitkile
rde düzelme olmadı, yem takviyesi yapmalı mıyız?
```

In [4]:
```
# [3] Cümlelere ayırma ve ilk 10 cümleyi gösterme
sentences = []
for text in texts:
    if isinstance(text, str):
        sentences.extend(sent_tokenize(text))
    else:
        continue
print("\nİlk 10 cümle:")
print(sentences[:10])
```

```
İlk 10 cümle:
['Bu yaz sezonunda şeftali tarlamızda yapraklarda kahverengileşme ve büzüşme.', 'Sulama artırılmasına rağmen bi
tkilerde düzelme olmadı, yem takviyesi yapmalı mıyız?', 'mısır tarlamızda genç fidelerde çekiş kaybı yaşanması.
', 'aktarma organlarında sorun kaynaklı olabilir mi?', 'patates tarlamızda genç fidelerde genç sürgünlerde kuru
malar ve yaprak dökümü.', 'azot eksikliği kaynaklı olabilir mi?', 'Bu yaz kurak geçtiği için sığırların genç sü
rgünlerde kurumalar ve yaprak dökümü.', 'gübreleme programını yeniden düzenlemeli miyim?', 'Son hasatta traktör
mizin çekiş kaybı yaşanması.', 'gübreleme programını yeniden düzenlemeli miyim?']
```

In [5]:
```
# [4] Türkçe stopwords listesini alma ve ilk 50'sini yazdırma
turkish_stopwords = set([
    've', 'ile', 'de', 'da', 'ki', 'kadar', 'için', 'ama', 'ya', 'veya',
    'bir', 'bu', 'şu', 'o', 'ne', 'nasıl', 'niye', 'hangi', 'her', 'tüm',
    'mi', 'mı', 'mu', 'mü', 'ise', 'değil', 'çok', 'az'
])
stop_words_list = list(turkish_stopwords)
print("\nTürkçe Stopwords (ilk 50):")
print(stop_words_list[:50])
```

```
Türkçe Stopwords (ilk 50):
['da', 'şu', 'ne', 'de', 'tüm', 'mu', 'o', 'ki', 'çok', 've', 'ama', 'hangi', 'bu', 'için', 'her', 'bir', 'az',
'ya', 'mı', 'veya', 'niye', 'ile', 'değil', 'kadar', 'ise', 'nasıl', 'mi', 'mü']
```

In [6]:
```
# [5] Stemleme fonksiyonunu başlatma
def simple_turkish_stem(token):
    suffixes = ['ler', 'lar', 'in', 'ın', 'un', 'ün', 'de', 'da', 'ki', 'e', 'a']
    for suffix in suffixes:
        if token.endswith(suffix):
            return token[:-len(suffix)]
    return token
```

In [7]:
```
# [6] Ön işleme fonksiyonunu tanımlama
def preprocess_sentence(sentence):
    if not isinstance(sentence, str):
        return [], []
    tokens = word_tokenize(sentence)
    filtered_tokens = [token.lower() for token in tokens if token.isalpha() and token.lower() not in turkish_st
    lemmatized_tokens = filtered_tokens  # Zembereksiz: filtrelenmiş kelimeler
    stemmed_tokens = [simple_turkish_stem(token) for token in filtered_tokens]
    return lemmatized_tokens, stemmed_tokens
```

In [8]:
```
# [7-8] Cümleleri işleme (fonksiyonlu)
tokenized_corpus_lemmatized = []
tokenized_corpus_stemmed = []
for sentence in sentences:
    lemmatized_tokens, stemmed_tokens = preprocess_sentence(sentence)
    tokenized_corpus_lemmatized.append(lemmatized_tokens)
```

```
    tokenized_corpus_stemmed.append(stemmed_tokens)
```

In [9]:
```
# İlk işlenen cümlenin çıktısını göster
print("\nİlk İşlenen Cümle (Fonksiyonlu):")
print(f"Ham: {sentences[0]}")
print(f"Lemmatized: {tokenized_corpus_lemmatized[0]}")
print(f"Stemmed: {tokenized_corpus_stemmed[0]}")
```

```
İlk İşlenen Cümle (Fonksiyonlu):
Ham: Bu yaz sezonunda şeftali tarlamızda yapraklarda kahverengileşme ve büzüşme.
Lemmatized: ['yaz', 'sezonunda', 'şeftali', 'tarlamızda', 'yapraklarda', 'kahverengileşme', 'büzüşme']
Stemmed: ['yaz', 'sezonun', 'şeftali', 'tarlamız', 'yapraklar', 'kahverengileşm', 'büzüşm']
```

In [10]:
```
# [9] Lemmatize edilmiş cümleleri CSV'ye kaydetme
with open("lemmatized_sentences.csv", mode="w", newline="", encoding="utf-8") as file:
    writer = csv.writer(file)
    for tokens in tokenized_corpus_lemmatized:
        writer.writerow([' '.join(tokens)])
```

In [11]:
```
# [10] Stemlenmiş cümleleri CSV'ye kaydetme
with open("stemmed_sentences.csv", mode="w", newline="", encoding="utf-8") as file:
    writer = csv.writer(file)
    for tokens in tokenized_corpus_stemmed:
        writer.writerow([' '.join(tokens)])
```

In [12]:
```
# [11] İlk 5 cümlenin ham, lemmatize ve stemlenmiş hallerini yazdırma
print("\nİlk 5 Cümle Karşılaştırması (Fonksiyonlu):")
for i in range(min(5, len(sentences))):
    print(f"Cümle {i+1} - Ham: {sentences[i]}")
    print(f"Cümle {i+1} - Lemmatized: {tokenized_corpus_lemmatized[i]}")
    print(f"Cümle {i+1} - Stemmed: {tokenized_corpus_stemmed[i]}")
    print()
```

```
İlk 5 Cümle Karşılaştırması (Fonksiyonlu):
Cümle 1 - Ham: Bu yaz sezonunda şeftali tarlamızda yapraklarda kahverengileşme ve büzüşme.
Cümle 1 - Lemmatized: ['yaz', 'sezonunda', 'şeftali', 'tarlamızda', 'yapraklarda', 'kahverengileşme', 'büzüşme'
]
Cümle 1 - Stemmed: ['yaz', 'sezonun', 'şeftali', 'tarlamız', 'yapraklar', 'kahverengileşm', 'büzüşm']

Cümle 2 - Ham: Sulama artırılmasına rağmen bitkilerde düzelme olmadı, yem takviyesi yapmalı mıyız?
Cümle 2 - Lemmatized: ['sulama', 'artırılmasına', 'rağmen', 'bitkilerde', 'düzelme', 'olmadı', 'yem', 'takviyes
i', 'yapmalı', 'mıyız']
Cümle 2 - Stemmed: ['sulam', 'artırılmasın', 'rağmen', 'bitkiler', 'düzelm', 'olmadı', 'yem', 'takviyesi', 'yap
malı', 'mıyız']

Cümle 3 - Ham: mısır tarlamızda genç fidelerde çekiş kaybı yaşanması.
Cümle 3 - Lemmatized: ['mısır', 'tarlamızda', 'genç', 'fidelerde', 'çekiş', 'kaybı', 'yaşanması']
Cümle 3 - Stemmed: ['mısır', 'tarlamız', 'genç', 'fideler', 'çekiş', 'kaybı', 'yaşanması']

Cümle 4 - Ham: aktarma organlarında sorun kaynaklı olabilir mi?
Cümle 4 - Lemmatized: ['aktarma', 'organlarında', 'sorun', 'kaynaklı', 'olabilir']
Cümle 4 - Stemmed: ['aktarm', 'organların', 'sor', 'kaynaklı', 'olabilir']

Cümle 5 - Ham: patates tarlamızda genç fidelerde genç sürgünlerde kurumalar ve yaprak dökümü.
Cümle 5 - Lemmatized: ['patates', 'tarlamızda', 'genç', 'fidelerde', 'genç', 'sürgünlerde', 'kurumalar', 'yapra
k', 'dökümü']
Cümle 5 - Stemmed: ['patates', 'tarlamız', 'genç', 'fideler', 'genç', 'sürgünler', 'kuruma', 'yaprak', 'dökümü'
]
```

In [44]:
```
# [12-15] Kütüphaneler, veri yükleme, cümlelere ayırma, stopwords
# Zaten [1-4]'te yapıldı.
```

In [15]:
```
# [16-17] Ayrıntılı for döngüsü ile kelimeleri tokenleştirme ve filtreleme
filtered_sentences = []
for sentence in sentences:
    tokens = word_tokenize(sentence)
    filtered_tokens = []
    for token in tokens:
        if token.isalpha():
            token_lower = token.lower()
            if token_lower not in turkish_stopwords:
                filtered_tokens.append(token_lower)
    filtered_sentences.append(filtered_tokens)
```

In [16]:
```
# [18] Lemmatize edilmiş cümleleri oluşturma (ayrıntılı, Zembereksiz)
tokenized_corpus_lemmatized_detailed = filtered_sentences
```

In [17]:
```
# [19] Ayrıntılı stemleme
tokenized_corpus_stemmed_detailed = []
for filtered_tokens in filtered_sentences:
    stemmed_tokens = [simple_turkish_stem(token) for token in filtered_tokens]
    tokenized_corpus_stemmed_detailed.append(stemmed_tokens)
print("\nAyrıntılı Stemlenmiş Cümleler (ilk 10):")
print(tokenized_corpus_stemmed_detailed[:10])
```

```
Ayrıntılı Stemlenmiş Cümleler (ilk 10):
[['yaz', 'sezonun', 'şeftali', 'tarlamız', 'yapraklar', 'kahverengileşm', 'büzüşm'], ['sulam', 'artırılmasın',
'rağmen', 'bitkiler', 'düzelm', 'olmadı', 'yem', 'takviyesi', 'yapmalı', 'mıyız'], ['mısır', 'tarlamız', 'genç'
, 'fideler', 'çekiş', 'kaybı', 'yaşanması'], ['aktarm', 'organların', 'sor', 'kaynaklı', 'olabilir'], ['patates
', 'tarlamız', 'genç', 'fideler', 'genç', 'sürgünler', 'kuruma', 'yaprak', 'dökümü'], ['azot', 'eksikliği', 'ka
ynaklı', 'olabilir'], ['yaz', 'kurak', 'geçtiği', 'sığırlar', 'genç', 'sürgünler', 'kuruma', 'yaprak', 'dökümü'
], ['gübrelem', 'programını', 'yeniden', 'düzenlemeli', 'miyim'], ['son', 'hasatt', 'traktörmiz', 'çekiş', 'kay
bı', 'yaşanması'], ['gübrelem', 'programını', 'yeniden', 'düzenlemeli', 'miyim']]
```

In [18]:
```python
# [20] Ayrıntılı stemlenmiş cümleleri CSV'ye kaydetme
with open("stemmed_sentences_detailed.csv", mode="w", newline="", encoding="utf-8") as file:
    writer = csv.writer(file)
    for tokens in tokenized_corpus_stemmed_detailed:
        writer.writerow([' '.join(tokens)])
```

In [19]:
```python
# [21] İlk 5 cümlenin ayrıntılı işlenmiş hallerini yazdırma
print("\nİlk 5 Cümle Karşılaştırması (Ayrıntılı):")
for i in range(min(5, len(sentences))):
    print(f"Cümle {i+1} - Ham: {sentences[i]}")
    print(f"Cümle {i+1} - Lemmatized: {tokenized_corpus_lemmatized_detailed[i]}")
    print(f"Cümle {i+1} - Stemmed: {tokenized_corpus_stemmed_detailed[i]}")
    print()
```

```
İlk 5 Cümle Karşılaştırması (Ayrıntılı):
Cümle 1 - Ham: Bu yaz sezonunda şeftali tarlamızda yapraklarda kahverengileşme ve büzüşme.
Cümle 1 - Lemmatized: ['yaz', 'sezonunda', 'şeftali', 'tarlamızda', 'yapraklarda', 'kahverengileşme', 'büzüşme'
]
Cümle 1 - Stemmed: ['yaz', 'sezonun', 'şeftali', 'tarlamız', 'yapraklar', 'kahverengileşm', 'büzüşm']

Cümle 2 - Ham: Sulama artırılmasına rağmen bitkilerde düzelme olmadı, yem takviyesi yapmalı mıyız?
Cümle 2 - Lemmatized: ['sulama', 'artırılmasına', 'rağmen', 'bitkilerde', 'düzelme', 'olmadı', 'yem', 'takviyes
i', 'yapmalı', 'mıyız']
Cümle 2 - Stemmed: ['sulam', 'artırılmasın', 'rağmen', 'bitkiler', 'düzelm', 'olmadı', 'yem', 'takviyesi', 'yap
malı', 'mıyız']

Cümle 3 - Ham: mısır tarlamızda genç fidelerde çekiş kaybı yaşanması.
Cümle 3 - Lemmatized: ['mısır', 'tarlamızda', 'genç', 'fidelerde', 'çekiş', 'kaybı', 'yaşanması']
Cümle 3 - Stemmed: ['mısır', 'tarlamız', 'genç', 'fideler', 'çekiş', 'kaybı', 'yaşanması']

Cümle 4 - Ham: aktarma organlarında sorun kaynaklı olabilir mi?
Cümle 4 - Lemmatized: ['aktarma', 'organlarında', 'sorun', 'kaynaklı', 'olabilir']
Cümle 4 - Stemmed: ['aktarm', 'organların', 'sor', 'kaynaklı', 'olabilir']

Cümle 5 - Ham: patates tarlamızda genç fidelerde genç sürgünlerde kurumalar ve yaprak dökümü.
Cümle 5 - Lemmatized: ['patates', 'tarlamızda', 'genç', 'fidelerde', 'genç', 'sürgünlerde', 'kurumalar', 'yapra
k', 'dökümü']
Cümle 5 - Stemmed: ['patates', 'tarlamız', 'genç', 'fideler', 'genç', 'sürgünler', 'kuruma', 'yaprak', 'dökümü'
]
```

In [45]:
```python
# 2. Hafta: TF-IDF Vektörizasyon
# [1] Ön işleme (zaten yapıldı, lemmatize verileri kullanacağım)

# [2-3] Lemmatize metinleri oluşturma ve ilk 3'ünü gösterme
lemmatized_texts = [' '.join(tokens) for tokens in tokenized_corpus_lemmatized]
print("\nİlk 3 lemmatized metin:")
print(lemmatized_texts[:3])
```

```
İlk 3 lemmatized metin:
['yaz sezonunda şeftali tarlamızda yapraklarda kahverengileşme büzüşme', 'sulama artırılmasına rağmen bitkilerd
e düzelme olmadı yem takviyesi yapmalı mıyız', 'mısır tarlamızda genç fidelerde çekiş kaybı yaşanması']
```

In [21]:
```python
# [4] TF-IDF vektörizasyon
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(lemmatized_texts)
feature_names = vectorizer.get_feature_names_out()
tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=feature_names)
```

In [22]:
```python
# [5] İlk 5 cümlenin TF-IDF skorlarını yazdırma
print("\nİlk 5 cümlenin TF-IDF skorları:")
print(tfidf_df.head())
```

```
İlk 5 cümlenin TF-IDF skorları:
   acil   aktarma  aldığımız  analizi  arttı  artırılmasına  atların  \
0   0.0  0.000000        0.0      0.0    0.0       0.000000      0.0
1   0.0  0.000000        0.0      0.0    0.0       0.312483      0.0
2   0.0  0.000000        0.0      0.0    0.0       0.000000      0.0
3   0.0  0.487473        0.0      0.0    0.0       0.000000      0.0
4   0.0  0.000000        0.0      0.0    0.0       0.000000      0.0

   azalması  azot  açmalı  ...  yumuşama  zeytin  zeytinlerde     çekiş  \
0       0.0   0.0     0.0  ...       0.0     0.0          0.0  0.000000
1       0.0   0.0     0.0  ...       0.0     0.0          0.0  0.000000
2       0.0   0.0     0.0  ...       0.0     0.0          0.0  0.398635
3       0.0   0.0     0.0  ...       0.0     0.0          0.0  0.000000
4       0.0   0.0     0.0  ...       0.0     0.0          0.0  0.000000

   çürüklüğü  çürümesi  üzüm  üzümlerde   şeftali  şeftalilerde
0        0.0       0.0   0.0        0.0  0.468997           0.0
1        0.0       0.0   0.0        0.0  0.000000           0.0
2        0.0       0.0   0.0        0.0  0.000000           0.0
3        0.0       0.0   0.0        0.0  0.000000           0.0
4        0.0       0.0   0.0        0.0  0.000000           0.0

[5 rows x 143 columns]
```

In [23]:
```python
# İlk cümle için TF-IDF skorlarını al
first_sentence_vector = tfidf_df.iloc[0]
top_5_words = first_sentence_vector.sort_values(ascending=False).head(5)
print("\nİlk cümlede en yüksek TF-IDF skoruna sahip 5 kelime:")
print(top_5_words)
```

```
İlk cümlede en yüksek TF-IDF skoruna sahip 5 kelime:
şeftali            0.468997
büzüşme            0.387385
kahverengileşme    0.387385
yapraklarda        0.387385
sezonunda          0.387385
Name: 0, dtype: float64
```

In [24]:
```python
# [6] "sorun" kelimesi için cosine benzerlik analizi
try:
    sorun_index = list(feature_names).index('sorun')
    sorun_vector = tfidf_matrix[:, sorun_index].toarray()
    tfidf_vectors = tfidf_matrix.toarray()
    similarities = cosine_similarity(sorun_vector.T, tfidf_vectors.T)
    similarities = similarities.flatten()
    top_5_indices = similarities.argsort()[-6:][:-1]
    print("\n'sorun' kelimesine en benzer 5 kelime:")
    for index in top_5_indices:
        print(f"{feature_names[index]}: {similarities[index]:.4f}")
except ValueError:
    print("\n'sorun' kelimesi veri setinde bulunamadı.")
```

```
'sorun' kelimesine en benzer 5 kelime:
durum: 0.1135
belirtisi: 0.1135
olabilir: 0.1747
organlarında: 1.0000
sorun: 1.0000
```

In [46]:
```python
# 3. Hafta: Word2Vec Model Eğitimi
# [16] Word2Vec modelleri eğitme
# Ön işleme zaten yapıldı, tokenized_corpus_lemmatized ve tokenized_corpus_stemmed kullanacağım

# Word2Vec modeli eğitmek için parametreler
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': 'cbow', 'window': 4, 'vector_size': 300},
    {'model_type': 'skipgram', 'window': 4, 'vector_size': 300}
]
```

In [47]:
```python
# Fonksiyon ile Word2Vec modeli eğitme ve kaydetme
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"
    print(f"{model_name}_{params['model_type']}_window{params['window']}_dim{params['vector_size']}.model saved
```

In [48]:
```python
from gensim.models import Word2Vec
```

In [49]:
```python
# Lemmatize edilmiş corpus ile modelleri eğitme ve kaydetme
for param in parameters:
    train_and_save_model(tokenized_corpus_lemmatized, param, "lemmatized_model")
    # Stemlenmiş corpus ile modelleri eğitme ve kaydetme
```

```
for param in parameters:
    train_and_save_model(tokenized_corpus_stemmed, param, "stemmed_model")
```

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

In [50]:

```
# [17] Üç model yükleme ve "sorun" kelimesi için en benzer 3 kelimeyi yazdırma
# Model dosyalarını yükleme
try:
    model_1 = Word2Vec.load("lemmatized_model_cbow_window2_dim100.model")
    model_2 = Word2Vec.load("stemmed_model_skipgram_window2_dim300.model")
    model_3 = Word2Vec.load("lemmatized_model_skipgram_window2_dim300.model")

    # 'sorun' kelimesi ile en benzer 3 kelimeyi ve skorlarını yazdırma
    def print_similar_words(model, model_name):
        try:
            similarity = model.wv.most_similar('sorun', topn=3)
            print(f"\n{model_name} Modeli - 'sorun' ile En Benzer 3 Kelime:")
            for word, score in similarity:
                print(f"Kelime: {word}, Benzerlik Skoru: {score:.4f}")
        except KeyError:
            print(f"\n{model_name} Modeli - 'sorun' kelimesi modelde bulunamadı.")

    # 3 model için benzer kelimeleri yazdırma
    print_similar_words(model_1, "Lemmatized CBOW Window 2 Dim 100")
    print_similar_words(model_2, "Stemmed Skipgram Window 2 Dim 300")
    print_similar_words(model_3, "Lemmatized Skipgram Window 2 Dim 300")
except FileNotFoundError:
    print("\nModellerden biri bulunamadı. Önce modelleri eğitip kaydedin.")
```

```
Lemmatized CBOW Window 2 Dim 100 Modeli - 'sorun' ile En Benzer 3 Kelime:
Kelime: düzenlemeli, Benzerlik Skoru: 0.3071
Kelime: gelişim, Benzerlik Skoru: 0.2873
Kelime: makinesimizin, Benzerlik Skoru: 0.2793

Stemmed Skipgram Window 2 Dim 300 Modeli - 'sorun' ile En Benzer 3 Kelime:
Kelime: tar, Benzerlik Skoru: 0.0686
Kelime: ver, Benzerlik Skoru: 0.0404
Kelime: gübr, Benzerlik Skoru: 0.0158

Lemmatized Skipgram Window 2 Dim 300 Modeli - 'sorun' ile En Benzer 3 Kelime:
Kelime: tarım, Benzerlik Skoru: 0.0686
Kelime: verim, Benzerlik Skoru: 0.0404
Kelime: gübre, Benzerlik Skoru: 0.0158
```

In [ ]: