# Hafta 10: Metin Madenciliği ve Duygu Analizi

## Giriş: Metin Madenciliği Nedir?

Metin madenciliği (Text Mining), yapılandırılmamış veya yarı yapılandırılmış metin verileri içerisinden anlamlı, kullanışlı ve önceki analizlerde ortaya çıkmamış bilgileri elde etmeyi amaçlayan disiplinler arası bir araştırma alanıdır. Doğal Dil İşleme (Natural Language Processing – NLP), makine öğrenmesi, istatistiksel analiz ve bilgi erişimi gibi alanlardan beslenen metin madenciliği, büyük hacimli dijital metin verilerinin işlenmesinde ve analizinde kritik bir rol oynar.

Günümüzde dijitalleşmenin etkisiyle metin tabanlı veri hacmi dramatik şekilde artmıştır. Haber siteleri, sosyal medya platformları, çevrimiçi kullanıcı yorumları, müşteri geri bildirimleri, çağrı merkezi kayıtları, e-posta içerikleri ve forum mesajları gibi kaynaklardan elde edilen veriler büyük ölçüde yapılandırılmamıştır. Bu tür verilerin klasik veri analiz teknikleriyle işlenmesi mümkün olmadığından, metin madenciliği teknikleri kullanılarak bu içeriklerden bilgi çıkarımı yapılması zorunlu hale gelmiştir.

Metin madenciliğinin temel hedeflerinden biri, doğal dil ile ifade edilen bilgiyi biçimsel ve analiz edilebilir bir yapıya dönüştürmektir. Bu süreç, metin önişleme (tokenization, stop-word removal, stemming/lemmatization), öznitelik çıkarımı (TF-IDF, word embeddings), metin temsili, sınıflandırma, kümeleme, konu modelleme (topic modeling), duygu analizi ve anlamsal benzerlik ölçümleri gibi çok sayıda adımdan oluşmaktadır.

Bu alandaki uygulama alanları son derece geniştir.

* **Pazarlama ve marka yönetimi:** Sosyal medya gönderileri ve müşteri yorumları analiz edilerek marka algısı ölçülebilir, ürün geliştirme süreçleri yönlendirilebilir.
* **Müşteri memnuniyeti analizi:** Otomatik anket değerlendirmesi ve çağrı merkezi kayıtlarından şikayet nedenleri analiz edilebilir.
* **Sahtekarlık tespiti:** Finansal belgeler, e-posta içerikleri ve müşteri taleplerinden anomali veya tutarsızlık tespitleri yapılabilir.
* **Kamuoyu araştırmaları:** Seçim dönemlerinde Twitter gibi sosyal medya kaynaklarından eğilim analizleri çıkarılabilir.
* **Yargı ve hukuk bilişimi:** Mahkeme kararları, sözleşmeler gibi yasal belgelerde metin madenciliği uygulamaları ile içerik sınıflandırması ve örüntü keşfi yapılabilir.

Metin madenciliği yalnızca bilgi keşfi açısından değil, aynı zamanda karar destek sistemlerinin geliştirilmesinde de stratejik bir rol üstlenmektedir. Artan veri hacmi ve çeşitliliği, bu alandaki tekniklerin sürekli olarak gelişmesini ve uygulama alanlarının genişlemesini zorunlu kılmaktadır.

## Metin Önişleme Teknikleri

Ham metin verileri doğrudan analiz edilebilecek nitelikte değildir. Bu nedenle, metin madenciliği ve doğal dil işleme (NLP) süreçlerinde, verilerin anlamlı bir yapıya dönüştürülmesi için bir dizi önişleme adımı gereklidir. Önişleme (preprocessing), metin içerisindeki anlamsal olmayan, tekrarlı, biçimsel ya da parazitik öğelerin temizlenmesini ve verinin daha az karmaşık hale getirilmesini sağlar. Bu işlemler, daha doğru, hızlı ve anlamlı sonuçlar elde edilmesini mümkün kılar. Aşağıda temel önişleme adımları açıklanmaktadır:

### 1. ****Küçük Harfe Çevirme (Lowercasing)****

Metin içerisindeki kelimeler genellikle büyük ve küçük harf karışık şekilde yazılır. Ancak doğal dil işleme algoritmaları büyük-küçük harf ayrımına duyarlıdır ve aynı kelimenin farklı biçimlerini (örneğin “Kitap” ve “kitap”) farklı terimler olarak değerlendirebilir. Bu nedenle tüm metnin küçük harfe dönüştürülmesi, kelime düzeyinde tutarlılık sağlar ve metriklerin doğru hesaplanmasına katkıda bulunur.

📌 Örnek: "Doğa Harikası" → "doğa harikası"

### 2. ****Noktalama İşaretlerinin ve Özel Karakterlerin Temizlenmesi (Punctuation & Special Character Removal)****

Noktalama işaretleri ve özel karakterler çoğu metin madenciliği uygulamasında anlamsal bir katkı sağlamaz. Ayrıca bu karakterlerin model tarafından kelime gibi algılanması istenmeyen girdilere yol açabilir. Bu nedenle, virgül, nokta, ünlem işareti, parantez gibi karakterler ve bazen sayılar metinden çıkarılır. Ancak bazı özel durumlarda (örneğin sosyal medya analizinde “#”, “@” gibi işaretler) bu karakterler korunabilir.

📌 Örnek: "Bugün hava çok güzel, değil mi?" → "Bugün hava çok güzel değil mi"

### 3. ****Tokenization (Metni Parçalara Ayırma)****

Tokenization, metnin kelime ya da cümle gibi daha küçük birimlere (token) bölünmesi işlemidir. Bu adım, metni bilgisayar tarafından işlenebilir hale getirir. En yaygın yaklaşım kelime bazlı tokenization’dır, ancak daha hassas uygulamalar için cümle ya da alt kelime düzeyinde tokenization da kullanılabilir.

📌 Örnek: "Bilim ve teknoloji hızla gelişiyor."  
Tokenlar: ["Bilim", "ve", "teknoloji", "hızla", "gelişiyor"]

### 4. ****Stopword Removal (Anlamsız Kelimelerin Çıkarılması)****

Stopword’ler, genellikle anlamsal olarak düşük bilgi içeriğine sahip, çok sık kullanılan kelimelerdir. Türkçe için “ve”, “ile”, “bir”, “ama”, “de” gibi bağlaçlar ve zamirler bu gruba girer. Analiz açısından anlam katmayan bu kelimelerin çıkarılması, işlem yükünü azaltır ve önemli kelimelerin daha baskın hale gelmesini sağlar.

📌 Örnek: "bu yüzden ve ancak yine de" gibi kelimeler genellikle çıkarılır.

### 5. ****Lemmatization / Stemming (Kök Bulma ve Temel Hâle Getirme)****

Bu adımda amaç, kelimenin kök ya da temel biçimine indirgenmesidir.

* **Stemming:** Kelimenin son eklerini keserek, anlamlı ya da anlamsız hale getirebilir. Daha hızlı ama daha kaba bir yöntemdir.  
  📌 Örnek: "koşuyorum" → "koş"
* **Lemmatization:** Kelimenin sözlükteki anlamlı hâline dönüştürülmesini sağlar. Dil bilgisel yapılar dikkate alınır. Daha yavaş fakat daha doğru sonuçlar verir.  
  📌 Örnek: "koşuyordu" → "koşmak"

Bu önişleme adımları, metnin analiz için daha sade ve anlamlı hale gelmesini sağlar. Özellikle sınıflandırma, duygu analizi, kümeleme gibi görevlerde bu işlemler, modelin başarımını doğrudan etkileyen kritik faktörlerdir.  
  
Python Örneği:  
import nltk  
from nltk.corpus import stopwords  
from nltk.tokenize import word\_tokenize  
from nltk.stem import WordNetLemmatizer  
import string  
  
text = "Bugün hava çok güzel ama biraz serin olabilir."  
tokens = word\_tokenize(text.lower())  
clean\_tokens = [word for word in tokens if word not in stopwords.words("turkish") and word not in string.punctuation]  
lemmatizer = WordNetLemmatizer()  
lemmas = [lemmatizer.lemmatize(token) for token in clean\_tokens]  
print("İşlenmiş Kelimeler:", lemmas)

## Duygu Analizi (Sentiment Analysis)

Duygu analizi, doğal dil işleme ve metin madenciliği alanlarında, metinlerdeki öznel ifadeleri sınıflandırmak, yorumlamak ve ölçmek amacıyla geliştirilen bir tekniktir. Temel amacı, bir metindeki duygusal yönelimi tespit etmek, yani ifadenin olumlu (positive), olumsuz (negative) veya nötr (neutral) bir duygu içerip içermediğini belirlemektir. Bu teknik, özellikle sosyal medya analizi, müşteri yorumları değerlendirmesi, pazar araştırması, siyasi eğilim tespiti ve kamuoyu yoklaması gibi pek çok uygulama alanında kullanılmaktadır.

Duygu analizinde iki temel yaklaşım öne çıkar:

### ****1. Sözlük (Lexicon)-Tabanlı Yaklaşım****

Sözlük tabanlı yaklaşımda, her kelimeye bir duygu değeri (genellikle -1 ile +1 arasında) atanır ve metindeki kelimelerin bu değerleri üzerinden toplam bir duygu skoru hesaplanır. Bu yöntem, kelimelerin anlamlarına ilişkin önceden tanımlanmış duygu sözlüklerine (sentiment lexicon) dayanır. Bazı yaygın kullanılan İngilizce sözlükler arasında **SentiWordNet**, **AFINN**, **VADER**, **LIWC** gibi kaynaklar bulunmaktadır. Türkçe için **Zemberek**, **TRSentiNet** gibi kaynaklar geliştirilmektedir.

Sözlük tabanlı yöntemler:

* **Yorumlama kolaylığı** sağlar, çünkü her kelimenin etkisi analiz edilebilir.
* Küçük veri kümeleri için **etiketsiz veri ile çalışabilme** avantajına sahiptir.
* Ancak deyim, ironi, bağlam farkı gibi durumları çoğu zaman **doğru değerlendiremez**.

📌 Örnek:  
Metin: “Film çok güzeldi ama sonu hayal kırıklığıydı.”  
Bu metindeki “güzel” kelimesi pozitif, “hayal kırıklığı” ise negatif olarak değerlendirilir. Toplam duygu skoru genellikle bağlama göre yorumlanır.

### ****2. Makine Öğrenmesi (Machine Learning)-Tabanlı Yaklaşım****

Bu yaklaşımda, sınıflandırma algoritmaları kullanılarak metinlerin otomatik olarak duygu etiketleri tahmin edilir. Genellikle etiketlenmiş (pozitif/negatif) bir eğitim veri setine ihtiyaç duyar. Naive Bayes, Destek Vektör Makineleri (SVM), Karar Ağaçları, Lojistik Regresyon gibi geleneksel yöntemler kullanılabilirken, son dönemde **derin öğrenme tabanlı** yöntemler (örneğin LSTM, BERT, Transformer modelleri) çok daha yüksek başarı oranları sağlamaktadır.

Makine öğrenmesi tabanlı yöntemler:

* **Daha esnek ve bağlamsal olarak duyarlı** sonuçlar üretir.
* Büyük ve çeşitli veri setleri ile çalışıldığında **yüksek doğruluk** sağlar.
* Ancak **etiketlenmiş veri gereksinimi** ve eğitim sürecinin karmaşıklığı bu yöntemin zorlukları arasındadır.

📌 Örnek:  
Tweet: “Bu telefonun kamerası şahane ama batarya rezalet.”  
Model, bu cümledeki olumlu ve olumsuz yönleri birlikte değerlendirip, genel duyguyu bağlama göre sınıflandırabilir.

### ****Karşılaştırmalı Değerlendirme****

| **Özellik** | **Sözlük Tabanlı** | **Makine Öğrenmesi Tabanlı** |
| --- | --- | --- |
| Veri ihtiyacı | Etiketsiz yeterlidir | Etiketlenmiş veri gerekir |
| Karmaşıklık | Düşük | Orta-Yüksek |
| Performans (genel) | Orta | Yüksek (özellikle büyük veri) |
| Bağlam Anlayışı | Sınırlı | Gelişmiş |
| Uygulama kolaylığı | Yüksek | Daha teknik bilgi gerektirir |

### ****Sonuç****

Duygu analizi, metin içerisindeki kullanıcı tutumunu anlamaya yönelik güçlü bir araçtır. Kullanılan yöntem, projenin veri yapısına, kaynaklara ve hedeflerine bağlı olarak değişir. Genellikle en başarılı sistemler, sözlük tabanlı ve makine öğrenmesi yaklaşımlarını hibrit şekilde birleştiren sistemlerdir.

Python ile TextBlob:  
from textblob import TextBlob  
  
yorum = TextBlob("Bu ürünü gerçekten çok beğendim, harika bir deneyim yaşadım!")  
print("Polarity:", yorum.sentiment.polarity)  
print("Subjectivity:", yorum.sentiment.subjectivity)

## Duygu Analizi Uygulaması (Naive Bayes ile)

Scikit-learn ile örnek sınıflandırma:  
from sklearn.feature\_extraction.text import CountVectorizer  
from sklearn.naive\_bayes import MultinomialNB  
from sklearn.model\_selection import train\_test\_split  
  
texts = ["Harika bir ürün", "Çok kötü, memnun kalmadım", "Muhteşem!", "Berbat bir deneyim"]  
labels = ["pozitif", "negatif", "pozitif", "negatif"]  
  
vectorizer = CountVectorizer()  
X = vectorizer.fit\_transform(texts)  
y = labels  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25)  
model = MultinomialNB()  
model.fit(X\_train, y\_train)  
print("Tahmin:", model.predict(X\_test))

## Görselleştirme

Duygu analizi sonuçları wordcloud, barplot veya pie chart şeklinde görselleştirilebilir.  
  
WordCloud Örneği:  
from wordcloud import WordCloud  
import matplotlib.pyplot as plt  
  
text = " ".join(lemmas)  
wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(text)  
plt.figure(figsize=(10, 5))  
plt.imshow(wordcloud, interpolation='bilinear')  
plt.axis('off')  
plt.show()