# 🎓 DOĞAL DİL İŞLEME - DETAYLI KONU ANLATIMI

## 🎯 1. MARKOV MODELİ - DETAYLI ANALİZ

### 📖 **Teorik Temeller**

#### **Tarihsel Gelişim:**

Markov modelleri, **Andrey Markov** tarafından 1906’da geliştirildi. İlk olarak Rus şiirindeki sesli harf dizilerini analiz etmek için kullanıldı.

#### **Matematiksel Temel:**

Markov özelliği, bir **stokastik süreç**’in gelecekteki durumunun **yalnızca mevcut duruma** bağlı olduğunu, geçmiş durumların etkisinin olmadığını belirtir.

**Formal Tanım:**

P(Xₙ₊₁ = x | X₁ = x₁, X₂ = x₂, ..., Xₙ = xₙ) = P(Xₙ₊₁ = x | Xₙ = xₙ)

### 🔢 **Dil Modellemede Markov Zinciri**

#### **N-Gram Modelleri:**

* **Unigram:** P(w₁, w₂, …, wₙ) = P(w₁) × P(w₂) × … × P(wₙ)
* **Bigram:** P(w₁, w₂, …, wₙ) = P(w₁) × P(w₂|w₁) × P(w₃|w₂) × … × P(wₙ|wₙ₋₁)
* **Trigram:** P(w₁, w₂, …, wₙ) = P(w₁) × P(w₂|w₁) × P(w₃|w₁,w₂) × … × P(wₙ|wₙ₋₂,wₙ₋₁)

#### **Detaylı Örnek - Bigram Model:**

**İngilizce Örnek:**

**Verilen Kelime Geçiş Olasılıkları:**

P(BAŞLANGIÇ|savaş) = 0.1  
P(savaş|tazminatı) = 0.3  
P(tazminatı|aldılar) = 0.2  
P(aldılar|.) = 0.8  
P(.|SON) = 0.9

**Cümle:** “Savaş tazminatı aldılar.”

**Adım Adım Hesaplama:**

P(cümle) = P(savaş|BAŞLANGIÇ) × P(tazminatı|savaş) × P(aldılar|tazminatı) × P(.|aldılar) × P(SON|.)  
  
P(cümle) = 0.1 × 0.3 × 0.2 × 0.8 × 0.9  
P(cümle) = 0.00432

**Türkçe Örnek:**

**Verilen Kelime Geçiş Olasılıkları:**

P(BAŞLANGIÇ|bugün) = 0.15  
P(bugün|hava) = 0.4  
P(hava|çok) = 0.3  
P(çok|güzel) = 0.6  
P(güzel|.) = 0.7  
P(.|SON) = 0.9

**Cümle:** “Bugün hava çok güzel.”

**Adım Adım Hesaplama:**

P(cümle) = P(bugün|BAŞLANGIÇ) × P(hava|bugün) × P(çok|hava) × P(güzel|çok) × P(.|güzel) × P(SON|.)  
  
P(cümle) = 0.15 × 0.4 × 0.3 × 0.6 × 0.7 × 0.9  
P(cümle) = 0.006804

**Karşılaştırmalı Analiz:**

| Özellik | İngilizce | Türkçe |
| --- | --- | --- |
| **Cümle** | “Savaş tazminatı aldılar.” | “Bugün hava çok güzel.” |
| **Uzunluk** | 4 kelime | 5 kelime |
| **Olasılık** | 0.00432 | 0.006804 |
| **Konu** | Tarihsel | Günlük |
| **Karmaşıklık** | Orta | Düşük |

### 📊 **Markov Modelinin Avantajları ve Dezavantajları**

#### ✅ **Avantajlar:**

* **Hesaplama Kolaylığı:** Sadece mevcut duruma bakılır
* **Bellek Verimliliği:** Geçmiş bilgiler saklanmaz
* **Hızlı Eğitim:** Az parametre ile eğitilir

#### ❌ **Dezavantajlar:**

* **Kısa Hafıza:** Uzun bağımlılıkları yakalayamaz
* **Sıfır Olasılık:** Eğitim verisinde görülmeyen geçişler
* **Bağlam Eksikliği:** Uzun mesafeli bağlamları unutur

### 🔧 **Smoothing Teknikleri**

#### **Laplace Smoothing (Add-1):**

P(wᵢ|wᵢ₋₁) = (count(wᵢ₋₁, wᵢ) + 1) / (count(wᵢ₋₁) + V)

V = kelime dağarcığı boyutu

#### **Interpolasyon:**

P(wᵢ|wᵢ₋₂, wᵢ₋₁) = λ₁P(wᵢ) + λ₂P(wᵢ|wᵢ₋₁) + λ₃P(wᵢ|wᵢ₋₂, wᵢ₋₁)

## 🎯 2. SHIFT-REDUCE ALGORİTMASI - DETAYLI ANALİZ

### 📖 **Teorik Temeller**

#### **Tarihsel Gelişim:**

Shift-Reduce algoritması, **Donald Knuth** tarafından 1965’te geliştirilen **LR parsing** algoritmasının temelidir. **Bottom-up parsing** yaklaşımını kullanır.

#### **Matematiksel Temel:**

Shift-Reduce, **yığın (stack)** tabanlı bir algoritmadır. Cümleyi **soldan sağa** işler ve **en sağdan türetme (rightmost derivation)** yapar.

**Temel İşlemler:** 1. **Shift:** Girdi kelimesini yığına ekle 2. **Reduce:** Yığındaki öğeleri gramer kuralına göre birleştir 3. **Accept:** Cümle başarıyla parse edildi 4. **Error:** Gramer hatası tespit edildi

### 🔢 **Algoritma Adımları - Detaylı Açıklama**

#### **Adım 1: Yığın ve Girdi Hazırlama**

Yığın: [] (boş)  
Girdi: [w₁, w₂, w₃, ..., wₙ]

#### **Adım 2: İşlem Döngüsü**

Her adımda şu kararlardan biri alınır: - **Shift:** Girdiden kelime al, yığına ekle - **Reduce:** Yığındaki öğeleri gramer kuralına göre birleştir - **Accept:** Cümle tamamlandı - **Error:** Gramer hatası

### 📝 **Detaylı Örnek - Adım Adım**

#### **İngilizce Örnek:**

**Gramer:**

S → NP VP  
NP → Det N  
VP → V NP  
Det → the  
N → cat, dog  
V → chased

**Cümle:** “the cat chased the dog”

**Adım Adım İşlem:**

| Adım | Yığın | Girdi | İşlem | Açıklama |
| --- | --- | --- | --- | --- |
| 1 | [] | [the, cat, chased, the, dog] | Shift | “the” yığına ekle |
| 2 | [the] | [cat, chased, the, dog] | Shift | “cat” yığına ekle |
| 3 | [the, cat] | [chased, the, dog] | Reduce | Det + N → NP |
| 4 | [NP] | [chased, the, dog] | Shift | “chased” yığına ekle |
| 5 | [NP, chased] | [the, dog] | Shift | “the” yığına ekle |
| 6 | [NP, chased, the] | [dog] | Shift | “dog” yığına ekle |
| 7 | [NP, chased, the, dog] | [] | Reduce | Det + N → NP |
| 8 | [NP, chased, NP] | [] | Reduce | V + NP → VP |
| 9 | [NP, VP] | [] | Reduce | NP + VP → S |
| 10 | [S] | [] | Accept | Cümle tamamlandı |

#### **Türkçe Örnek:**

**Gramer:**

S → NP VP  
NP → Det N  
VP → V NP  
Det → bu, şu, o  
N → kedi, köpek, kuş  
V → kovaladı, gördü

**Cümle:** “bu kedi köpeği kovaladı”

**Adım Adım İşlem:**

| Adım | Yığın | Girdi | İşlem | Açıklama |
| --- | --- | --- | --- | --- |
| 1 | [] | [bu, kedi, köpeği, kovaladı] | Shift | “bu” yığına ekle |
| 2 | [bu] | [kedi, köpeği, kovaladı] | Shift | “kedi” yığına ekle |
| 3 | [bu, kedi] | [köpeği, kovaladı] | Reduce | Det + N → NP |
| 4 | [NP] | [köpeği, kovaladı] | Shift | “köpeği” yığına ekle |
| 5 | [NP, köpeği] | [kovaladı] | Shift | “kovaladı” yığına ekle |
| 6 | [NP, köpeği, kovaladı] | [] | Reduce | N + V → VP |
| 7 | [NP, VP] | [] | Reduce | NP + VP → S |
| 8 | [S] | [] | Accept | Cümle tamamlandı |

#### **Karşılaştırmalı Analiz:**

| Özellik | İngilizce | Türkçe |
| --- | --- | --- |
| **Cümle** | “the cat chased the dog” | “bu kedi köpeği kovaladı” |
| **Uzunluk** | 5 kelime | 4 kelime |
| **Adım Sayısı** | 10 adım | 8 adım |
| **Yapı** | Det N V Det N | Det N N V |
| **Karmaşıklık** | Orta | Düşük |
| **Reduce İşlemleri** | 3 adet | 2 adet |

### 📊 **Shift-Reduce Algoritmasının Avantajları ve Dezavantajları**

#### ✅ **Avantajlar:**

* **Hızlı İşlem:** O(n) zaman karmaşıklığı
* **Bellek Verimli:** Sadece yığın boyutu kadar bellek
* **Gerçek Zamanlı:** Kelime kelime işleyebilir
* **Hata Tespiti:** İlk hatada durur
* **Esnek:** Herhangi bir gramer türü ile çalışır

#### ❌ **Dezavantajlar:**

* **Karmaşık Karar:** Shift mi Reduce mu yapılacağına karar vermek zor
* **Ambiguity:** Birden fazla parse ağacı olabilir
* **Gramer Bağımlılığı:** Gramer kalitesi önemli
* **Hata Kurtarma:** Hata sonrası devam etmek zor

### 🔧 **Karar Mekanizması**

#### **Shift-Reduce Kararı:**

Algoritma her adımda şu kriterlere bakar: 1. **Yığındaki son öğeler** gramer kuralına uyuyor mu? 2. **Girdideki sonraki kelime** hangi kuralı bekliyor? 3. **Lookahead:** Bir sonraki kelimeye bakarak karar ver

#### **Örnek Karar Süreci:**

Yığın: [NP, chased]  
Girdi: [the, dog]  
Durum: chased (V) var, NP bekliyor  
Karar: Shift (çünkü NP oluşturmak için daha fazla kelime gerekli)

### ⚡ **Algoritma Karmaşıklığı**

#### **Zaman Karmaşıklığı:**

* **O(n)** - Her kelime bir kez işlenir
* **En kötü durum:** O(n²) - Her adımda reduce

#### **Uzay Karmaşıklığı:**

* **O(n)** - Yığın boyutu cümle uzunluğu kadar

#### **Pratik Performans:**

n = 20 kelimelik cümle için:  
- Zaman: ~20-40 işlem  
- Uzay: ~20 kelime

## 🎯 3. CYK ALGORİTMASI - DETAYLI ANALİZ

### 📖 **Teorik Temeller**

#### **Tarihsel Gelişim:**

CYK (Cocke-Younger-Kasami) algoritması, **John Cocke**, **Daniel Younger** ve **Tadao Kasami** tarafından 1960’larda bağımsız olarak geliştirildi.

#### **Matematiksel Temel:**

CYK, **Chomsky Normal Form** (CNF) gramerlerle çalışan **dinamik programlama** algoritmasıdır.

**CNF Kuralları:** 1. A → BC (iki terminal olmayan) 2. A → a (tek terminal) 3. S → ε (sadece başlangıç sembolü için)

### 🔢 **Algoritma Adımları - Detaylı Açıklama**

#### **Adım 1: Grameri CNF’ye Dönüştür**

Orijinal Gramer:  
S → NP VP  
NP → Det N | N  
VP → V NP | V NP PP  
PP → P NP  
Det → the, a  
N → cat, dog, telescope  
V → saw, chased  
P → with  
  
CNF Dönüşümü:  
S → NP VP  
NP → Det N  
NP → N  
VP → V NP  
VP → V NP\_PP  
NP\_PP → NP PP  
PP → P NP  
Det → the  
Det → a  
N → cat  
N → dog  
N → telescope  
V → saw  
V → chased  
P → with

#### **Adım 2: Tablo Oluştur**

n×n matris oluştur (n = cümle uzunluğu)

#### **Adım 3: Köşegen Doldur (Tek Kelimeler)**

Cümle: "the cat chased the dog"  
Uzunluk: 5  
  
 1 2 3 4 5  
1 Det - - - -  
2 - N - - -  
3 - - V - -  
4 - - - Det -  
5 - - - - N

#### **Adım 4: Üst Köşegenleri Doldur**

**2’li Gruplar (Uzunluk 2):**

[1,2]: Det + N = NP  
[2,3]: N + V = (geçersiz)  
[3,4]: V + Det = (geçersiz)  
[4,5]: Det + N = NP  
  
 1 2 3 4 5  
1 Det NP - - -  
2 - N - - -  
3 - - V - -  
4 - - - Det NP  
5 - - - - N

**3’lü Gruplar (Uzunluk 3):**

[1,3]: Det + N + V = (geçersiz)  
[2,4]: N + V + Det = (geçersiz)  
[3,5]: V + Det + N = VP  
  
 1 2 3 4 5  
1 Det NP - - -  
2 - N - - -  
3 - - V - -  
4 - - - Det NP  
5 - - - - N

**4’lü Gruplar (Uzunluk 4):**

[1,4]: Det + N + V + Det = (geçersiz)  
[2,5]: N + V + Det + N = (geçersiz)

**5’li Gruplar (Uzunluk 5):**

[1,5]: Det + N + V + Det + N = S  
  
 1 2 3 4 5  
1 Det NP - - S  
2 - N - - -  
3 - - V - -  
4 - - - Det NP  
5 - - - - N

### 📝 **Detaylı Örnek - Adım Adım**

#### **İngilizce Örnek:**

**Gramer:**

S → NP VP  
NP → Det N  
VP → V NP  
Det → the  
N → cat, dog  
V → chased

**Cümle:** “the cat chased the dog”

**Tablo Doldurma Süreci:**

1. **Köşegen (Tek Kelimeler):**
   * [1,1]: “the” → Det
   * [2,2]: “cat” → N
   * [3,3]: “chased” → V
   * [4,4]: “the” → Det
   * [5,5]: “dog” → N
2. **2’li Gruplar:**
3. **3’lü Gruplar:**
4. **5’li Grup:**

**Sonuç:** S köşesinde S varsa cümle gramer tarafından üretilir.

#### **Türkçe Örnek:**

**Gramer:**

S → NP VP  
NP → Det N  
VP → V NP  
Det → bu, şu, o  
N → kedi, köpek, kuş  
V → kovaladı, gördü

**Cümle:** “bu kedi köpeği kovaladı”

**Tablo Doldurma Süreci:**

1. **Köşegen (Tek Kelimeler):**
   * [1,1]: “bu” → Det
   * [2,2]: “kedi” → N
   * [3,3]: “köpeği” → N
   * [4,4]: “kovaladı” → V
2. **2’li Gruplar:**
3. **4’lü Grup:**

**Sonuç:** S köşesinde S varsa cümle gramer tarafından üretilir.

#### **Karşılaştırmalı Tablo:**

| Özellik | İngilizce | Türkçe |
| --- | --- | --- |
| **Cümle** | “the cat chased the dog” | “bu kedi köpeği kovaladı” |
| **Uzunluk** | 5 kelime | 4 kelime |
| **Yapı** | Det N V Det N | Det N N V |
| **Karmaşıklık** | 5×5 tablo | 4×4 tablo |
| **İşlem Sayısı** | 10 adım | 8 adım |

### ⚡ **Algoritma Karmaşıklığı**

#### **Zaman Karmaşıklığı:**

* **O(n³)** - Üç iç içe döngü
* **Uzay Karmaşıklığı:** O(n²) - n×n tablo

#### **Pratik Uygulamalar:**

n = 10 kelimelik cümle için:  
- Zaman: 10³ = 1000 işlem  
- Uzay: 10² = 100 hücre

## 🎯 4. SHIFT-REDUCE vs CYK - DETAYLI KARŞILAŞTIRMA

### 📊 **Temel Karşılaştırma Tablosu**

| Özellik | Shift-Reduce | CYK |
| --- | --- | --- |
| **Yöntem** | Yığın tabanlı | Tablo doldurma |
| **Parsing Türü** | Bottom-up | Bottom-up |
| **Gramer Türü** | Herhangi | CNF |
| **Zaman Karmaşıklığı** | O(n) | O(n³) |
| **Uzay Karmaşıklığı** | O(n) | O(n²) |
| **Gerçek Zamanlı** | ✅ Evet | ❌ Hayır |
| **Hata Tespiti** | Hızlı | Yavaş |
| **Ambiguity** | İlk bulduğunu alır | Tüm parse ağaçlarını bulur |
| **Uygulama** | Pratik | Teorik |
| **Bellek Kullanımı** | Düşük | Yüksek |

### 🔄 **Detaylı Karşılaştırma**

#### **1. Çalışma Prensibi:**

**Shift-Reduce:**

- Soldan sağa işler  
- Yığın kullanır  
- Her adımda karar verir  
- İlk hatada durur

**CYK:**

- Tüm cümleyi görür  
- Tablo doldurur  
- Sonunda karar verir  
- Tüm olasılıkları hesaplar

#### **2. Gramer Gereksinimleri:**

**Shift-Reduce:**

- Herhangi bir gramer türü  
- Dönüştürme gerekmez  
- Esnek yapı

**CYK:**

- Sadece CNF gramerler  
- Dönüştürme zorunlu  
- Kısıtlı yapı

#### **3. Hata Yönetimi:**

**Shift-Reduce:**

- İlk hatada durur  
- Hızlı hata tespiti  
- Hata kurtarma zor

**CYK:**

- Tüm tabloyu doldurur  
- Yavaş hata tespiti  
- Hata analizi kolay

### 📝 **Pratik Örneklerle Karşılaştırma**

#### **Örnek 1: “the cat chased the dog”**

**Shift-Reduce Süreci:**

Adım 1: [the] ← Shift  
Adım 2: [the, cat] ← Shift   
Adım 3: [NP] ← Reduce (Det + N → NP)  
Adım 4: [NP, chased] ← Shift  
Adım 5: [NP, chased, the] ← Shift  
Adım 6: [NP, chased, the, dog] ← Shift  
Adım 7: [NP, chased, NP] ← Reduce (Det + N → NP)  
Adım 8: [NP, VP] ← Reduce (V + NP → VP)  
Adım 9: [S] ← Reduce (NP + VP → S)  
Adım 10: Accept

**CYK Süreci:**

1. Tablo oluştur (5×5)  
2. Köşegen doldur: [Det, N, V, Det, N]  
3. 2'li gruplar: [NP, -, -, NP]  
4. 3'lü gruplar: [-, -, VP, -]  
5. 5'li grup: [S]  
6. Sonuç: S köşesinde S var → Kabul

#### **Örnek 2: Ambiguous Cümle**

**Cümle:** “I saw the man with the telescope”

**Shift-Reduce:**

- İlk bulduğu parse ağacını kabul eder  
- Hızlı karar verir  
- Diğer olasılıkları görmez

**CYK:**

- Tüm olası parse ağaçlarını bulur  
- Ambiguity'yi tespit eder  
- Hangi yapının daha olası olduğunu hesaplar

### ⚡ **Performans Karşılaştırması**

#### **Zaman Karşılaştırması:**

Cümle Uzunluğu | Shift-Reduce | CYK  
10 kelime | ~10-20 işlem | ~1000 işlem  
20 kelime | ~20-40 işlem | ~8000 işlem  
50 kelime | ~50-100 işlem| ~125000 işlem

#### **Bellek Karşılaştırması:**

Cümle Uzunluğu | Shift-Reduce | CYK  
10 kelime | ~10 kelime | ~100 hücre  
20 kelime | ~20 kelime | ~400 hücre  
50 kelime | ~50 kelime | ~2500 hücre

### 🎯 **Hangi Durumda Hangi Algoritma?**

#### **Shift-Reduce Kullan:**

* **Gerçek zamanlı** uygulamalar
* **Hızlı** parsing gerektiğinde
* **Bellek** kısıtlı olduğunda
* **Pratik** uygulamalar
* **Hata tespiti** önemli olduğunda

#### **CYK Kullan:**

* **Teorik** analiz
* **Ambiguity** tespiti
* **Tüm olasılıkları** görmek istediğinde
* **Gramer doğrulama**
* **Akademik** çalışmalar

### 📊 **Gramer Dönüştürme Örneği**

#### **Normal Gramer → CNF:**

**Orijinal Gramer:**

S → NP VP  
NP → Det N | N  
VP → V NP | V NP PP  
PP → P NP  
Det → the, a  
N → cat, dog, telescope  
V → saw, chased  
P → with

**CNF Dönüşümü:**

S → NP VP  
NP → Det N  
NP → N  
VP → V NP  
VP → V NP\_PP  
NP\_PP → NP PP  
PP → P NP  
Det → the  
Det → a  
N → cat  
N → dog  
N → telescope  
V → saw  
V → chased  
P → with

**Dönüştürme Kuralları:** 1. **A → BCD** → **A → BE, E → CD** 2. **A → a|b** → **A → a, A → b** 3. **Terminal olmayanlar** sadece 2 terminal olmayan üretebilir

## 🎯 5. EXPECTATION (BEKLENTİ) - DETAYLI ANALİZ

### 📖 **Teorik Temeller**

#### **Matematiksel Tanım:**

Beklenti, bir rastgele değişkenin **ortalama değeri**’dir.

**Formal Tanım:**

E[X] = ∑ xᵢ × P(xᵢ)

#### **Sürekli Değişkenler İçin:**

E[X] = ∫ x × f(x) dx

### 🔢 **Markov Modelinde Beklenti**

#### **Durum Geçiş Beklentisi:**

Bir durumdan diğerine geçişin beklenen sayısı.

**Formül:**

E[Geçiş] = ∑ P(geçişᵢ) × değerᵢ

#### **Detaylı Örnek - Hava Durumu Modeli:**

**Geçiş Olasılıkları:**

Güneşli → Yağmurlu: 0.3  
Güneşli → Bulutlu: 0.5  
Güneşli → Güneşli: 0.2

**Beklenti Hesaplama:**

E[Geçiş] = 0.3 × 1 + 0.5 × 1 + 0.2 × 1 = 1

**Açıklama:** Her geçiş 1 adım sayıldığı için toplam beklenti 1’dir.

### 📊 **Dil Modellemede Beklenti**

#### **Kelime Beklentisi:**

Bir kelimenin bir cümlede kaç kez geçeceğinin beklenen değeri.

**Örnek:**

P("the") = 0.05  
Cümle uzunluğu = 20  
  
E["the"] = 20 × 0.05 = 1

#### **Cümle Uzunluğu Beklentisi:**

Bir dil modelinin üreteceği cümlelerin ortalama uzunluğu.

**Formül:**

E[Uzunluk] = ∑ i × P(Uzunluk = i)

### 🔧 **Koşullu Beklenti**

#### **Tanım:**

Bir olayın gerçekleşmesi koşuluyla başka bir olayın beklenen değeri.

**Formül:**

E[X|Y] = ∑ x × P(X=x|Y)

#### **Markov Modelinde:**

E[SonrakiKelime|MevcutKelime] = ∑ kelime × P(SonrakiKelime=kelime|MevcutKelime)

### 📝 **Pratik Uygulamalar**

#### **1. Metin Üretimi:**

Verilen: "Bugün hava"  
Beklenti: E[SonrakiKelime] = 0.3×"güzel" + 0.4×"kötü" + 0.3×"yağmurlu"  
Sonuç: "kötü" (en yüksek olasılık)

#### **2. Model Değerlendirmesi:**

Test cümlesi: "the cat sat on the mat"  
Model tahminleri:  
- P("cat"|"the") = 0.1  
- P("sat"|"cat") = 0.05  
- P("on"|"sat") = 0.2  
- P("the"|"on") = 0.3  
- P("mat"|"the") = 0.15  
  
Beklenti: E[Olasılık] = 0.1 × 0.05 × 0.2 × 0.3 × 0.15 = 0.000045

### ⚡ **Beklenti Hesaplama Teknikleri**

#### **1. Monte Carlo Simülasyonu:**

1. Çok sayıda rastgele örnek üret  
2. Her örneğin değerini hesapla  
3. Ortalamasını al

#### **2. Dinamik Programlama:**

1. Alt problemleri çöz  
2. Sonuçları tabloda sakla  
3. Büyük problemi küçük parçalardan oluştur

### 📊 **Beklenti vs Varyans**

#### **Beklenti (E[X]):**

* Ortalama değer
* Merkezi eğilim

#### **Varyans (Var[X]):**

Var[X] = E[(X - E[X])²] = E[X²] - (E[X])²

* Değişkenlik ölçüsü
* Dağılımın genişliği

### 🔄 **Markov Modelinde Varyans**

#### **Durum Geçiş Varyansı:**

Var[Geçiş] = E[Geçiş²] - (E[Geçiş])²

**Örnek:**

Geçiş değerleri: [1, 1, 1] (her geçiş 1 adım)  
Olasılıklar: [0.3, 0.5, 0.2]  
  
E[Geçiş] = 1  
E[Geçiş²] = 0.3×1² + 0.5×1² + 0.2×1² = 1  
Var[Geçiş] = 1 - 1² = 0

## 🎯 SINAVDA ÇIKABİLECEK SORU TİPLERİ

### 📝 **Markov Modeli Soruları:**

#### **Soru Tipi 1: Cümle Olasılığı Hesaplama**

Verilen: Geçiş olasılıkları ve bir cümle  
İstenen: Cümlenin olasılığını hesapla  
  
Çözüm: P(w₁...wₙ) = P(w₁) × ∏P(wᵢ|wᵢ₋₁)

#### **Soru Tipi 2: Beklenti Hesaplama**

Verilen: Durum geçiş olasılıkları  
İstenen: Beklenen geçiş sayısını hesapla  
  
Çözüm: E[X] = ∑xᵢ × P(xᵢ)

### 📝 **CYK Algoritması Soruları:**

#### **Soru Tipi 1: Tablo Doldurma**

Verilen: CNF gramer ve cümle  
İstenen: CYK tablosunu doldur  
  
Çözüm: Köşegenden başla, üst köşegenleri doldur

#### **Soru Tipi 2: Gramer Dönüştürme**

Verilen: Normal gramer  
İstenen: CNF'ye dönüştür  
  
Çözüm: A → BCD → A → BE, E → CD

### 📝 **Shift-Reduce Soruları:**

#### **Soru Tipi 1: Adım Adım Parsing**

Verilen: Gramer ve cümle  
İstenen: Shift-Reduce adımlarını göster  
  
Çözüm: Yığın ve girdi durumlarını takip et

#### **Soru Tipi 2: Karar Mekanizması**

Verilen: Yığın ve girdi durumu  
İstenen: Shift mi Reduce mu yapılacağına karar ver  
  
Çözüm: Gramer kurallarına ve lookahead'e bak

### 📝 **Expectation Soruları:**

#### **Soru Tipi 1: Koşullu Beklenti**

Verilen: Koşullu olasılıklar  
İstenen: E[X|Y] hesapla  
  
Çözüm: E[X|Y] = ∑x × P(X=x|Y)

#### **Soru Tipi 2: Model Değerlendirmesi**

Verilen: Test verisi ve model tahminleri  
İstenen: Beklenen performansı hesapla  
  
Çözüm: E[Performans] = ∑P(tahmin) × değer

## 🎓 BAŞARI TAVSİYELERİ

### ✅ **Son Gün Çalışma Planı:**

1. **Markov Modeli (2 saat):**
   * Formülleri ezberle
   * Örnek problemler çöz
   * Beklenti hesaplamaları yap
2. **CYK Algoritması (2 saat):**
   * Tablo doldurma pratiği
   * CNF dönüşümleri
   * Shift-Reduce karşılaştırması
3. **Shift-Reduce (1 saat):**
   * Adım adım parsing
   * Karar mekanizması
   * Pratik örnekler
4. **Expectation (1 saat):**
   * Koşullu beklenti hesaplamaları
   * Varyans formülleri
   * Pratik uygulamalar

### 🎯 **Sınav İpuçları:**

* **Zaman yönetimi:** CYK en çok zaman alır
* **Hesaplama kontrolü:** Sonuçları mantık kontrolünden geçir
* **Birim kontrolü:** Olasılıklar 0-1 arasında olmalı
* **Grafik çiz:** CYK için tablo çiz
* **Adım takibi:** Shift-Reduce için yığın durumunu takip et

### 📝 **Formül Kartı:**

Markov: P(w₁...wₙ) = P(w₁) × ∏P(wᵢ|wᵢ₋₁)  
CYK: O(n³) zaman, O(n²) uzay  
Shift-Reduce: O(n) zaman, O(n) uzay  
Expectation: E[X] = ∑xᵢ × P(xᵢ)  
Koşullu: E[X|Y] = ∑x × P(X=x|Y)  
Varyans: Var[X] = E[X²] - (E[X])²

**🎓 BAŞARILAR! Final sınavında başarılar dilerim! 🎓**

*Bu döküman, CYK, Markov, Shift-Reduce ve Expectation konularına özel olarak detaylandırılmıştır.*

### 📝 **Pratik Örnek - Spam Tespiti**

**Verilen:** - 1000 e-posta test edildi - 200 spam, 800 normal - Sonuçlar: - 150 spam doğru tespit edildi - 50 spam yanlış tespit edildi (normal sanıldı) - 100 normal yanlış tespit edildi (spam sanıldı) - 700 normal doğru tespit edildi

**Karmaşıklık Matrisi:**

Tahmin  
Gerçek Spam Normal  
Spam 150 50  
Normal 100 700

**Hesaplamalar:** - TP = 150 (Spam doğru tespit) - FP = 100 (Normal yanlış spam) - TN = 700 (Normal doğru tespit) - FN = 50 (Spam yanlış normal)

### 📝 **Türkçe Örnek - Duygu Analizi**

**Verilen:** - 500 yorum test edildi - 300 pozitif, 200 negatif - Sonuçlar: - 250 pozitif doğru tespit edildi - 50 pozitif yanlış tespit edildi (negatif sanıldı) - 30 negatif yanlış tespit edildi (pozitif sanıldı) - 170 negatif doğru tespit edildi

**Karmaşıklık Matrisi:**

Tahmin  
Gerçek Pozitif Negatif  
Pozitif 250 50  
Negatif 30 170

**Hesaplamalar:** - TP = 250 (Pozitif doğru tespit) - FP = 30 (Negatif yanlış pozitif) - TN = 170 (Negatif doğru tespit) - FN = 50 (Pozitif yanlış negatif)

### 📝 **İngilizce Örnek - Email Classification**

**Given:** - 1200 emails tested - 400 work, 800 personal - Results: - 350 work emails correctly classified - 50 work emails misclassified as personal - 80 personal emails misclassified as work - 720 personal emails correctly classified

**Confusion Matrix:**

Prediction  
Actual Work Personal  
Work 350 50  
Personal 80 720

**Calculations:** - TP = 350 (Work correctly classified) - FP = 80 (Personal misclassified as work) - TN = 720 (Personal correctly classified) - FN = 50 (Work misclassified as personal)

### 📖 **Öklid Mesafesi (Euclidean Distance)**

**Formül:**

d(x⃗, y⃗) = √(∑ᵢ₌₁ⁿ (xᵢ - yᵢ)²)

**İngilizce Pratik Örnek:**

Doküman A: [1, 2, 3, 0, 1] (the, cat, sat, on, mat)  
Doküman B: [2, 1, 3, 1, 0] (the, dog, sat, on, floor)  
  
d(A,B) = √((1-2)² + (2-1)² + (3-3)² + (0-1)² + (1-0)²)  
d(A,B) = √(1 + 1 + 0 + 1 + 1) = √4 = 2

**Türkçe Pratik Örnek:**

Doküman A: [2, 1, 0, 1, 1] (kedi, evde, uyuyor, çok, mutlu)  
Doküman B: [1, 2, 1, 0, 1] (kedi, bahçede, oynuyor, çok, mutlu)  
  
d(A,B) = √((2-1)² + (1-2)² + (0-1)² + (1-0)² + (1-1)²)  
d(A,B) = √(1 + 1 + 1 + 1 + 0) = √4 = 2

### 📖 **Kosinüs Benzerliği (Cosine Similarity)**

**Formül:**

cos(θ) = (x⃗ · y⃗) / (||x⃗|| × ||y⃗||)

**İngilizce Pratik Örnek:**

Doküman A: [1, 2, 3, 0, 1] (the, cat, sat, on, mat)  
Doküman B: [2, 1, 3, 1, 0] (the, dog, sat, on, floor)  
  
A · B = 1×2 + 2×1 + 3×3 + 0×1 + 1×0 = 2 + 2 + 9 + 0 + 0 = 13  
||A|| = √(1² + 2² + 3² + 0² + 1²) = √15  
||B|| = √(2² + 1² + 3² + 1² + 0²) = √15  
  
cos(θ) = 13 / (√15 × √15) = 13/15 ≈ 0.867

**Türkçe Pratik Örnek:**

Doküman A: [2, 1, 0, 1, 1] (kedi, evde, uyuyor, çok, mutlu)  
Doküman B: [1, 2, 1, 0, 1] (kedi, bahçede, oynuyor, çok, mutlu)  
  
A · B = 2×1 + 1×2 + 0×1 + 1×0 + 1×1 = 2 + 2 + 0 + 0 + 1 = 5  
||A|| = √(2² + 1² + 0² + 1² + 1²) = √7  
||B|| = √(1² + 2² + 1² + 0² + 1²) = √7  
  
cos(θ) = 5 / (√7 × √7) = 5/7 ≈ 0.714

### ⚠️ **Farklar**

* **Öklid:** Mutlak mesafe, küçük değer = benzer
* **Kosinüs:** Açı benzerliği, büyük değer = benzer
* **Kosinüs:** Doküman uzunluğundan etkilenmez

### 📊 **Karşılaştırmalı Analiz:**

| Özellik | İngilizce | Türkçe |
| --- | --- | --- |
| **Öklid Mesafesi** | 2.0 | 2.0 |
| **Kosinüs Benzerliği** | 0.867 | 0.714 |
| **Doküman Uzunluğu** | 5 kelime | 5 kelime |
| **Ortak Kelimeler** | 3 (the, sat, on) | 3 (kedi, çok, mutlu) |
| **Benzerlik Seviyesi** | Yüksek | Orta |

## 🎯 MATRİS YERLEŞİMİ - ADIM ADIM REHBER

### 📊 1. KARMAŞIKLIK MATRİSİ (CONFUSION MATRIX) YERLEŞİMİ

#### **Adım 1: Matris Boyutunu Belirle**

Sınıf sayısı = n ise → n×n matris  
İkili sınıflandırma için: 2×2 matris

#### **Adım 2: Satır ve Sütun Başlıklarını Yerleştir**

Tahmin (Prediction)  
Gerçek (Actual) Sınıf1 Sınıf2  
Sınıf1 TP FN  
Sınıf2 FP TN

#### **Adım 3: Hücreleri Doldur**

TP (True Positive): Doğru pozitif tahminler  
FP (False Positive): Yanlış pozitif tahminler   
TN (True Negative): Doğru negatif tahminler  
FN (False Negative): Yanlış negatif tahminler

#### **Adım 4: Pratik Örnek - Adım Adım**

**Verilen:** - 1000 e-posta test edildi - 200 spam, 800 normal - Sonuçlar: - 150 spam doğru tespit edildi - 50 spam yanlış tespit edildi (normal sanıldı) - 100 normal yanlış tespit edildi (spam sanıldı) - 700 normal doğru tespit edildi

**Adım 1: Matris Yapısını Çiz**

Tahmin  
Gerçek Spam Normal  
Spam [ ] [ ]  
Normal [ ] [ ]

**Adım 2: TP ve FN Hücrelerini Doldur (Spam Satırı)**

TP = 150 (Spam doğru tespit)  
FN = 50 (Spam yanlış normal)  
  
 Tahmin  
Gerçek Spam Normal  
Spam 150 50  
Normal [ ] [ ]

**Adım 3: FP ve TN Hücrelerini Doldur (Normal Satırı)**

FP = 100 (Normal yanlış spam)  
TN = 700 (Normal doğru tespit)  
  
 Tahmin  
Gerçek Spam Normal  
Spam 150 50  
Normal 100 700

**Adım 4: Kontrol Et**

Toplam Spam: 150 + 50 = 200 ✓  
Toplam Normal: 100 + 700 = 800 ✓  
Toplam Test: 200 + 800 = 1000 ✓

### 📊 2. CYK TABLOSU YERLEŞİMİ

#### **Adım 1: Cümle Uzunluğunu Belirle**

Cümle: "the cat chased the dog"  
Uzunluk: 5 kelime → 5×5 tablo

#### **Adım 2: Tablo Yapısını Oluştur**

1 2 3 4 5  
1 [ ] [ ] [ ] [ ] [ ]  
2 [ ] [ ] [ ] [ ] [ ]  
3 [ ] [ ] [ ] [ ] [ ]  
4 [ ] [ ] [ ] [ ] [ ]  
5 [ ] [ ] [ ] [ ] [ ]

#### **Adım 3: Köşegeni Doldur (Tek Kelimeler)**

[1,1]: "the" → Det  
[2,2]: "cat" → N  
[3,3]: "chased" → V  
[4,4]: "the" → Det  
[5,5]: "dog" → N  
  
 1 2 3 4 5  
1 Det [ ] [ ] [ ] [ ]  
2 [ ] N [ ] [ ] [ ]  
3 [ ] [ ] V [ ] [ ]  
4 [ ] [ ] [ ] Det [ ]  
5 [ ] [ ] [ ] [ ] N

#### **Adım 4: 2’li Grupları Doldur (Uzunluk 2)**

[1,2]: Det + N → NP  
[2,3]: N + V → (geçersiz)  
[3,4]: V + Det → (geçersiz)  
[4,5]: Det + N → NP  
  
 1 2 3 4 5  
1 Det NP [ ] [ ] [ ]  
2 [ ] N [ ] [ ] [ ]  
3 [ ] [ ] V [ ] [ ]  
4 [ ] [ ] [ ] Det NP  
5 [ ] [ ] [ ] [ ] N

#### **Adım 5: 3’lü Grupları Doldur (Uzunluk 3)**

[1,3]: Det + N + V → (geçersiz)  
[2,4]: N + V + Det → (geçersiz)  
[3,5]: V + Det + N → VP  
  
 1 2 3 4 5  
1 Det NP [ ] [ ] [ ]  
2 [ ] N [ ] [ ] [ ]  
3 [ ] [ ] V [ ] [ ]  
4 [ ] [ ] [ ] Det NP  
5 [ ] [ ] [ ] [ ] N

#### **Adım 6: 4’lü Grupları Doldur (Uzunluk 4)**

[1,4]: Det + N + V + Det → (geçersiz)  
[2,5]: N + V + Det + N → (geçersiz)

#### **Adım 7: 5’li Grubu Doldur (Uzunluk 5)**

[1,5]: Det + N + V + Det + N → S  
  
 1 2 3 4 5  
1 Det NP [ ] [ ] S  
2 [ ] N [ ] [ ] [ ]  
3 [ ] [ ] V [ ] [ ]  
4 [ ] [ ] [ ] Det NP  
5 [ ] [ ] [ ] [ ] N

### 📊 3. TÜRKÇE CYK TABLOSU ÖRNEĞİ

#### **Cümle:** “bu kedi köpeği kovaladı”

#### **Adım 1: 4×4 Tablo Oluştur**

1 2 3 4  
1 [ ] [ ] [ ] [ ]  
2 [ ] [ ] [ ] [ ]  
3 [ ] [ ] [ ] [ ]  
4 [ ] [ ] [ ] [ ]

#### **Adım 2: Köşegeni Doldur**

[1,1]: "bu" → Det  
[2,2]: "kedi" → N  
[3,3]: "köpeği" → N  
[4,4]: "kovaladı" → V  
  
 1 2 3 4  
1 Det [ ] [ ] [ ]  
2 [ ] N [ ] [ ]  
3 [ ] [ ] N [ ]  
4 [ ] [ ] [ ] V

#### **Adım 3: 2’li Grupları Doldur**

[1,2]: Det + N → NP  
[2,3]: N + N → (geçersiz)  
[3,4]: N + V → VP  
  
 1 2 3 4  
1 Det NP [ ] [ ]  
2 [ ] N [ ] [ ]  
3 [ ] [ ] N [ ]  
4 [ ] [ ] [ ] V

#### **Adım 4: 3’lü Grupları Doldur**

[1,3]: Det + N + N → (geçersiz)  
[2,4]: N + N + V → (geçersiz)

#### **Adım 5: 4’lü Grubu Doldur**

[1,4]: Det + N + N + V → S  
  
 1 2 3 4  
1 Det NP [ ] S  
2 [ ] N [ ] [ ]  
3 [ ] [ ] N [ ]  
4 [ ] [ ] [ ] V

### 📊 4. MATRİS YERLEŞİM KURALLARI

#### **Karmaşıklık Matrisi Kuralları:**

1. **Satırlar:** Gerçek değerler
2. **Sütunlar:** Tahmin edilen değerler
3. **Köşegen:** Doğru tahminler (TP, TN)
4. **Köşegen dışı:** Yanlış tahminler (FP, FN)
5. **Satır toplamları:** Gerçek sınıf sayıları
6. **Sütun toplamları:** Tahmin edilen sınıf sayıları

#### **CYK Tablosu Kuralları:**

1. **Boyut:** n×n (n = cümle uzunluğu)
2. **Köşegen:** Tek kelimeler (uzunluk 1)
3. **Üst köşegenler:** Çoklu kelime grupları
4. **[i,j] hücresi:** i’den j’ye kadar olan kelimeler
5. **Doldurma sırası:** Köşegenden başla, yukarı çık
6. **Son kontrol:** [1,n] hücresinde S var mı?

#### **Genel Matris Kuralları:**

1. **Satır numaraları:** Soldan sağa
2. **Sütun numaraları:** Yukarıdan aşağı
3. **Hücre koordinatları:** [satır, sütun]
4. **Boş hücreler:** “-” veya boş bırak
5. **Kontrol:** Toplamlar tutarlı olmalı

### 📊 5. SINAV İPUÇLARI

#### **Karmaşıklık Matrisi:**

* **Önce yapıyı çiz** (satır/sütun başlıkları)
* **TP ve TN’yi köşegene** yerleştir
* **FP ve FN’yi köşegen dışına** yerleştir
* **Toplamları kontrol et**

#### **CYK Tablosu:**

* **Önce boyutu belirle** (cümle uzunluğu)
* **Köşegeni doldur** (tek kelimeler)
* **Yukarı çık** (2’li, 3’lü, 4’lü gruplar)
* **Gramer kurallarını** uygula
* **Son hücreyi kontrol et** ([1,n])

#### **Hata Kontrolü:**

* **Matris boyutu** doğru mu?
* **Toplamlar** tutarlı mı?
* **Gramer kuralları** uygulandı mı?
* **Sonuç** mantıklı mı?

**🎓 BAŞARILAR! Matris yerleşiminde dikkatli olun! 🎓**

## 🎯 SLAYT KONULARI - DETAYLI ANALİZ

### 📊 1. ZIPF YASASI - DETAYLI AÇIKLAMA

#### **📖 Teorik Temel:**

Zipf yasası, **George Kingsley Zipf** tarafından 1935’te keşfedilen, dildeki kelime frekansları ile sıralamaları arasındaki matematiksel ilişkiyi açıklayan bir yasadır.

#### **🔢 Matematiksel Formül:**

Zipf = R × F / N\_toplam

* **R:** Kelimenin sıralaması
* **F:** Kelimenin frekansı
* **N\_toplam:** Toplam kelime sayısı

#### **📝 Slayttaki Örnek:**

N\_toplam = 90800  
"ile" kelimesi: R = 2, F = 676  
Zipf = 2 × 676 / 90800 = 0.022  
  
"ile" kelimesi: R = 6, F = 511   
Zipf = 6 × 511 / 90800 = 0.033

#### **🎯 Pratik Uygulama:**

**Türkçe Örnek:**

Toplam kelime sayısı: 50000  
"bir" kelimesi: 1. sırada, 2500 kez geçiyor  
"ve" kelimesi: 2. sırada, 1800 kez geçiyor  
  
Zipf("bir") = 1 × 2500 / 50000 = 0.05  
Zipf("ve") = 2 × 1800 / 50000 = 0.072

#### **⚠️ Önemli Noktalar:**

* **Enerji korunumu:** En sık kullanılan kelimeler az sayıda
* **Doğal sıralama:** Yeni kelimeler eklendiğinde sıralama değişir
* **Sabit oran:** Frekans ile sıralama arasında sabit ilişki

### 📊 2. CÜMLE OLASILIKLARI - DETAYLI AÇIKLAMA

#### **📖 Teorik Temel:**

Bir cümlenin olasılığı, cümleyi oluşturan kelimelerin olasılıklarının çarpımıdır.

#### **🔢 Matematiksel Formül:**

P(w₁, w₂, ..., wₙ) = P(w₁) × P(w₂) × ... × P(wₙ)

#### **📝 Slayttaki Örnek:**

P(w = bir) = frekans(bir) / toplam = 3180/10900 = 0.292

#### **🎯 Pratik Uygulama:**

**Türkçe Örnek:**

"bugün hava güzel" cümlesi için:  
P(bugün) = 150/5000 = 0.03  
P(hava) = 200/5000 = 0.04   
P(güzel) = 100/5000 = 0.02  
  
P(cümle) = 0.03 × 0.04 × 0.02 = 0.000024

**İngilizce Örnek:**

"the cat sat" cümlesi için:  
P(the) = 500/3000 = 0.167  
P(cat) = 50/3000 = 0.017  
P(sat) = 30/3000 = 0.01  
  
P(cümle) = 0.167 × 0.017 × 0.01 = 0.000028

#### **⚠️ Önemli Noktalar:**

* **Bağımsızlık varsayımı:** Kelimeler birbirinden bağımsız
* **Sıra önemsiz:** “yüz oldu” = “oldu yüz”
* **Çok küçük değerler:** Uzun cümleler için çok küçük olasılıklar

### 📊 3. ENTROPİ - DETAYLI AÇIKLAMA

#### **📖 Teorik Temel:**

Entropi, bir sistemdeki belirsizliği veya rastgeleliği ölçen bir kavramdır. Bilgi teorisinde, bir mesajın taşıdığı bilgi miktarını ölçmek için kullanılır.

#### **🔢 Matematiksel Formül:**

H(x) = ∑ P(x) × log(1/P(x))

#### **📝 Slayttaki Örnek:**

AVM'ye gelen arabalar:  
P(sedan) = 100/200 = 0.5  
P(hatchback) = 50/200 = 0.25  
P(station) = 25/200 = 0.125  
P(sport) = 25/200 = 0.125  
  
H(x) = 0.5×1 + 0.25×2 + 2×0.125×3 = 1.5

#### **🎯 Pratik Uygulama:**

**Türkçe Örnek - Hava Durumu:**

P(güneşli) = 0.4  
P(yağmurlu) = 0.3  
P(bulutlu) = 0.2  
P(karlı) = 0.1  
  
H(x) = 0.4×1.32 + 0.3×1.74 + 0.2×2.32 + 0.1×3.32 = 1.85

**İngilizce Örnek - Email Türleri:**

P(work) = 0.5  
P(personal) = 0.3  
P(spam) = 0.2  
  
H(x) = 0.5×1 + 0.3×1.74 + 0.2×2.32 = 1.49

#### **⚠️ Önemli Noktalar:**

* **Maksimum entropi:** Eşit olasılıklar için
* **Minimum entropi:** Tek olasılık için (0)
* **Bilgi ölçümü:** Yüksek entropi = çok bilgi

### 📊 4. PERPLEXITY - DETAYLI AÇIKLAMA

#### **📖 Teorik Temel:**

Perplexity, bir dil modelinin ne kadar iyi çalıştığını ölçen bir metriktir. Düşük perplexity değeri, modelin daha iyi olduğunu gösterir.

#### **🔢 Matematiksel Formül:**

Perplexity = 2^H(x)

H(x) = Cross-entropy

#### **📝 Slayttaki Açıklama:**

* **Büyük metin kümesi** yerine **model olasılıkları** kullanılır
* **Gerçek dünyaya yakınsama** sağlanır
* **Model performansı** ölçülür

#### **🎯 Pratik Uygulama:**

**Türkçe Örnek:**

Model tahminleri:  
P(bugün|hava) = 0.3  
P(hava|güzel) = 0.4  
P(güzel|.) = 0.2  
  
Cross-entropy = -(log(0.3) + log(0.4) + log(0.2)) / 3 = 1.74  
Perplexity = 2^1.74 = 3.35

**İngilizce Örnek:**

Model tahminleri:  
P(the|cat) = 0.1  
P(cat|sat) = 0.05  
P(sat|.) = 0.2  
  
Cross-entropy = -(log(0.1) + log(0.05) + log(0.2)) / 3 = 3.32  
Perplexity = 2^3.32 = 10.0

### 📊 5. DİL MODELLERİ - DETAYLI AÇIKLAMA

#### **📖 Teorik Temel:**

Dil modelleri, kelime dizilerinin olasılıklarını hesaplayan modellerdir. Markov özelliği kullanarak, bir kelimenin olasılığı sadece önceki kelimelere bağlıdır.

#### **🔢 Matematiksel Formül:**

P(w₁, w₂, ..., wₙ) = P(w₁|BAŞLANGIÇ) × P(w₂|w₁) × P(w₃|w₂) × ... × P(wₙ|wₙ₋₁)

#### **📝 Slayttaki Örnek:**

"Savaş tazminatı aldılar." cümlesi için:  
P(cümle) = P(savaş|BAŞLANGIÇ) × P(tazminatı|savaş) × P(aldılar|tazminatı) × P(.|aldılar)

#### **🎯 Pratik Uygulama:**

**Türkçe Örnek:**

"Bugün hava çok güzel" cümlesi için:  
P(bugün|BAŞLANGIÇ) = 0.1  
P(hava|bugün) = 0.3  
P(çok|hava) = 0.2  
P(güzel|çok) = 0.4  
P(.|güzel) = 0.8  
  
P(cümle) = 0.1 × 0.3 × 0.2 × 0.4 × 0.8 = 0.00192

**İngilizce Örnek:**

"The cat sat on mat" cümlesi için:  
P(the|BAŞLANGIÇ) = 0.2  
P(cat|the) = 0.1  
P(sat|cat) = 0.05  
P(on|sat) = 0.3  
P(mat|on) = 0.2  
P(.|mat) = 0.9  
  
P(cümle) = 0.2 × 0.1 × 0.05 × 0.3 × 0.2 × 0.9 = 0.000054

#### **⚠️ Önemli Noktalar:**

* **Markov özelliği:** Sadece önceki kelimeye bağlı
* **Koşullu olasılık:** P(wᵢ|wᵢ₋₁) hesaplama
* **Smoothing:** Sıfır olasılık problemini çözme

### 📊 6. EŞDİZİMLİLİK (COLLOCATION) - DETAYLI AÇIKLAMA

#### **📖 Teorik Temel:**

Eşdizimlilik, kelimelerin birlikte sık geçmesi durumudur. Bu kelimeler anlamlı bir bütün oluşturur.

#### **🔢 Pointwise Mutual Information (PMI):**

I(x,y) = log₂(P(x,y) / (P(x) × P(y)))

#### **📝 Slayttaki Örnek:**

"New York" kelimesi için:  
P(New) = 541/1500 = 0.36  
P(York) = 212/1500 = 0.14  
P(New York) = 5/1500 = 0.003  
  
PMI = log₂(0.003 / (0.36 × 0.14)) = log₂(0.003 / 0.0504) = log₂(0.0595) = -4.07

#### **🎯 Pratik Uygulama:**

**Türkçe Örnek:**

"zaman kaybı" kelimesi için:  
P(zaman) = 200/1000 = 0.2  
P(kaybı) = 50/1000 = 0.05  
P(zaman kaybı) = 15/1000 = 0.015  
  
PMI = log₂(0.015 / (0.2 × 0.05)) = log₂(0.015 / 0.01) = log₂(1.5) = 0.585

**İngilizce Örnek:**

"fast food" kelimesi için:  
P(fast) = 150/2000 = 0.075  
P(food) = 300/2000 = 0.15  
P(fast food) = 25/2000 = 0.0125  
  
PMI = log₂(0.0125 / (0.075 × 0.15)) = log₂(0.0125 / 0.01125) = log₂(1.11) = 0.15

#### **⚠️ Null Hipotez Testi:**

H₀: P(x) × P(y) > P(x,y) → Bağımsız (eşdizim değil)  
H₁: P(x,y) > P(x) × P(y) → Eşdizim

### 📊 7. INTERPOLASYON - DETAYLI AÇIKLAMA

#### **📖 Teorik Temel:**

Interpolasyon, seyrek geçen kelimeler için smoothing tekniğidir. Farklı n-gram modellerinin ağırlıklı ortalamasını alır.

#### **🔢 Matematiksel Formül:**

P(wₙ|wₙ₋₂,wₙ₋₁) = λ₁P(wₙ) + λ₂P(wₙ|wₙ₋₁) + λ₃P(wₙ|wₙ₋₂,wₙ₋₁)

#### **📝 Slayttaki Açıklama:**

* **Sıfır olasılık problemi** çözülür
* **Lambda katsayıları** pozitif değerler
* **Farklı n-gram seviyeleri** birleştirilir

#### **🎯 Pratik Uygulama:**

**Türkçe Örnek:**

"zamazingo" kelimesi için:  
P(zamazingo) = 0 (hiç geçmemiş)  
P(zamazingo|önceki) = 0.001  
P(zamazingo|önceki,önceki) = 0.0001  
  
λ₁ = 0.1, λ₂ = 0.3, λ₃ = 0.6  
  
P(zamazingo|önceki,önceki) = 0.1×0 + 0.3×0.001 + 0.6×0.0001 = 0.00036

**İngilizce Örnek:**

"supercalifragilistic" kelimesi için:  
P(supercalifragilistic) = 0  
P(supercalifragilistic|previous) = 0.0005  
P(supercalifragilistic|prev,prev) = 0.0001  
  
λ₁ = 0.2, λ₂ = 0.4, λ₃ = 0.4  
  
P(supercalifragilistic|prev,prev) = 0.2×0 + 0.4×0.0005 + 0.4×0.0001 = 0.00024

### 📊 8. KELİME TORBASI (BAG OF WORDS) - DETAYLI AÇIKLAMA

#### **📖 Teorik Temel:**

Kelime torbası, bir dokümanı kelime frekanslarına göre vektör olarak temsil eden yaklaşımdır. Kelimelerin sırası önemsizdir.

#### **🔢 İşlem Adımları:**

1. **Kelime sınırlarının bulunması**
2. **Eklerin kaldırılması (stemming)**
3. **Sözlük oluşturma**
4. **Doküman vektörü oluşturma**

#### **📝 Slayttaki Örnek:**

Doküman: "Edirne'de Meriç, Tunca ve Arda nehirleri etrafında 2315 parça bakımlı sebze, meyve ve dut bahçeleri vardır."  
  
Kelime Çıkarımı: [edirne, meriç, tunca, arda, nehirleri, etrafında, 2315, parça, bakımlı, sebze, meyve, ve, dut, bahçeleri, vardır]  
  
Sözlük: {edirne, meriç, tunca, arda, nehir, etraf, parça, bakım, sebze, meyve, dut, yunanistan, savaş tazminatı, lozan antlaşması, karaağaç, türkiye, lozan anıtı, ilçe, inşa, ve, dan, ile, diğer, ol, edilen, aldı, mahale, var, bol, al, katıl, 15 Eylül 1923, 2315}

#### **🎯 Pratik Uygulama:**

**Türkçe Örnek:**

Doküman: "İstanbul'da hava bugün çok güzel. Boğaz manzarası muhteşem."  
  
Kelime Çıkarımı: [istanbul, da, hava, bugün, çok, güzel, boğaz, manzara, muhteşem]  
  
Sözlük: {istanbul, hava, bugün, çok, güzel, boğaz, manzara, muhteşem, ...}  
  
Doküman Vektörü: [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, ...]

**İngilizce Örnek:**

Document: "The weather in London is beautiful today. The Thames looks amazing."  
  
Word Extraction: [the, weather, in, london, is, beautiful, today, thames, looks, amazing]  
  
Dictionary: {the, weather, in, london, is, beautiful, today, thames, looks, amazing, ...}  
  
Document Vector: [2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, ...]

#### **⚠️ Önemli Noktalar:**

* **Sıra önemsiz:** Kelimelerin dizilimi önemli değil
* **Frekans önemli:** Aynı kelime birden fazla geçebilir
* **Boyut problemi:** Büyük sözlükler için seyrek vektörler

### 📊 9. DOKÜMAN VEKTÖRÜ - DETAYLI AÇIKLAMA

#### **📖 Teorik Temel:**

Doküman vektörü, bir dokümanı sayısal olarak temsil eden vektördür. Her boyut bir kelimeyi temsil eder.

#### **🔢 Vektör Türleri:**

1. **Geçip/geçmeme:** 1 veya 0
2. **Geçme sayısı:** Rakamla
3. **Ağırlık:** TF-IDF ile

#### **📝 Slayttaki Örnek:**

Doküman: "Edirne'de Meriç, Tunca ve Arda nehirleri etrafında 2315 parça bakımlı sebze, meyve ve dut bahçeleri vardır."  
  
Geçip/geçmeme durumu: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, ..., 1, 0, 0, 0, 1]  
  
Geçme sayısı: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 0, 0, 0, 0, 2, ..., 1, 0, 0, 0, 1]

#### **🎯 Pratik Uygulama:**

**Türkçe Örnek:**

Doküman: "Kedi evde uyuyor. Kedi çok mutlu."  
  
Sözlük: [kedi, evde, uyuyor, çok, mutlu, köpek, bahçede, oynuyor]  
  
Vektör (geçip/geçmeme): [1, 1, 1, 1, 1, 0, 0, 0]  
Vektör (geçme sayısı): [2, 1, 1, 1, 1, 0, 0, 0]

**İngilizce Örnek:**

Document: "The cat sat on the mat. The cat is happy."  
  
Dictionary: [the, cat, sat, on, mat, is, happy, dog, ran]  
  
Vector (binary): [1, 1, 1, 1, 1, 1, 1, 0, 0]  
Vector (count): [2, 2, 1, 1, 1, 1, 1, 0, 0]

#### **⚠️ Doküman Uzunluğu:**

Norm hesaplama: ||x|| = √(x₁² + x₂² + ... + xₙ²)  
  
Örnek: [1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 0, 0, 2, 2, 1, 1, ..., 1, 0, 3, 5, 1]  
Norm = √(1² + 1² + 1² + 1² + 2² + 1² + ... + 1²) = √(14 + 58 + 9) = 9

### 📊 10. SINIFLANDIRMA TÜRLERİ - DETAYLI AÇIKLAMA

#### **📖 Teorik Temel:**

Metin sınıflandırma, dokümanları belirli kategorilere atama işlemidir.

#### **🔢 Sınıflandırma Türleri:**

**1. İkili Sınıflandırma (Binary Classification):**

Örnek: Spam/Normal e-posta  
- 2 sınıftan sadece biri  
- P(sınıf1) + P(sınıf2) = 1

**2. Çok Sınıflı Sınıflandırma (Multiclass Classification):**

Örnek: Spor, Politika, Ekonomi, Yaşam  
- Birden fazla sınıftan sadece biri  
- P(sınıf1) + P(sınıf2) + ... + P(sınıfN) = 1

**3. Çok Etiketli Sınıflandırma (Multilabel Classification):**

Örnek: Bir doküman hem Spor hem Ekonomi olabilir  
- Birden fazla sınıfa aynı anda ait olabilir  
- Her sınıf için ayrı olasılık

#### **🎯 Pratik Uygulama:**

**Türkçe Örnekler:**

İkili: Spam/Normal SMS  
Çok Sınıflı: Haber kategorileri (Spor, Politika, Ekonomi)  
Çok Etiketli: Duygu analizi (Mutlu, Üzgün, Kızgın aynı anda)

**İngilizce Örnekler:**

Binary: Work/Personal email  
Multiclass: News categories (Sports, Politics, Economy)  
Multilabel: Sentiment analysis (Happy, Sad, Angry simultaneously)

### 📊 11. K-EN YAKIN KOMŞU (K-NN) - DETAYLI AÇIKLAMA

#### **📖 Teorik Temel:**

K-NN, bir dokümanı en yakın k komşusuna bakarak sınıflandıran algoritmadır.

#### **🔢 Algoritma Adımları:**

1. **Mesafe hesaplama** (Öklid, Kosinüs)
2. **En yakın k komşu bulma**
3. **Çoğunluk oylaması**

#### **📝 Slayttaki Açıklama:**

* **K değeri:** En yakın k adet komşu
* **Yoğunluk hesaplama:** Komşular arasında en çok hangi sınıf
* **Merkez yaklaşımı:** Sınıf merkez vektörüne en yakın

#### **🎯 Pratik Uygulama:**

**Türkçe Örnek:**

Yeni doküman: [1, 0, 1, 1, 0, 1, 0, 0]  
K = 3  
  
En yakın 3 komşu:  
- Komşu 1: Spor sınıfı, mesafe = 1.5  
- Komşu 2: Spor sınıfı, mesafe = 2.1   
- Komşu 3: Politika sınıfı, mesafe = 2.3  
  
Sonuç: Spor sınıfı (2/3 oy)

**İngilizce Örnek:**

New document: [1, 1, 0, 1, 0, 1, 0, 0]  
K = 5  
  
Nearest 5 neighbors:  
- Neighbor 1: Sports class, distance = 1.2  
- Neighbor 2: Sports class, distance = 1.8  
- Neighbor 3: Politics class, distance = 2.0  
- Neighbor 4: Sports class, distance = 2.1  
- Neighbor 5: Economy class, distance = 2.5  
  
Result: Sports class (3/5 votes)

#### **⚠️ Merkez Yaklaşımı:**

Her sınıf için ortalama vektör hesapla  
Yeni dokümanı en yakın merkeze ata  
  
Spor merkezi: [0.8, 0.2, 0.9, 0.1, 0.3, 0.7, 0.1, 0.2]  
Politika merkezi: [0.1, 0.9, 0.2, 0.8, 0.7, 0.1, 0.8, 0.1]

### 📊 12. DEĞERLENDİRME ÖLÇÜTLERİ - DETAYLI AÇIKLAMA

#### **📖 Teorik Temel:**

Sınıflandırma performansını ölçmek için kullanılan metriklerdir.

#### **🔢 Temel Metrikler:**

**1. Doğruluk (Accuracy):**

Accuracy = (TP + TN) / (TP + TN + FP + FN)

**2. Kesinlik (Precision):**

Precision = TP / (TP + FP)

**3. Duyarlılık (Recall):**

Recall = TP / (TP + FN)

**4. F-Ölçüsü (F-Measure):**

F-Measure = 2 × (Precision × Recall) / (Precision + Recall)

#### **📝 Slayttaki Örnek:**

1000 SMS: 200 Normal, 800 Spam  
Normal: 100 doğru, 100 yanlış  
Spam: 300 doğru, 500 yanlış  
  
Ham ortalama = (100/200 + 300/800) / 2 = 0.4375 = 43.75%  
Ağırlıklı ortalama = (200/1000 × 100/200) + (800/1000 × 300/800) = 0.4 = 40%

#### **🎯 Pratik Uygulama:**

**Türkçe Örnek:**

500 yorum: 300 pozitif, 200 negatif  
Pozitif: 250 doğru, 50 yanlış  
Negatif: 170 doğru, 30 yanlış  
  
TP = 250, FP = 30, TN = 170, FN = 50  
  
Accuracy = (250 + 170) / 500 = 84%  
Precision = 250 / (250 + 30) = 89.3%  
Recall = 250 / (250 + 50) = 83.3%  
F-Measure = 2 × (89.3 × 83.3) / (89.3 + 83.3) = 86.2%

**İngilizce Örnek:**

1200 emails: 400 work, 800 personal  
Work: 350 doğru, 50 yanlış  
Personal: 720 doğru, 80 yanlış  
  
TP = 350, FP = 80, TN = 720, FN = 50  
  
Accuracy = (350 + 720) / 1200 = 89.2%  
Precision = 350 / (350 + 80) = 81.4%  
Recall = 350 / (350 + 50) = 87.5%  
F-Measure = 2 × (81.4 × 87.5) / (81.4 + 87.5) = 84.3%

#### **⚠️ Ortalama Hesaplama:**

**Ham Ortalama:**

(Sınıf1\_başarı + Sınıf2\_başarı + ...) / Sınıf\_sayısı

**Ağırlıklı Ortalama:**

∑(Sınıf\_oranı × Başarı\_oranı)

**🎓 BAŞARILAR! Slayt konularını detaylıca öğrendin! 🎓**

*Bu bölüm, slaytlardaki tüm konuları detaylıca ele alır ve pratik uygulamaları gösterir.*