

İlk Dil Modelinizi Oluřturun

Dil Modelleri iin Uygulamalı Rehber

Hasan Degismez

2025-12-26

Dil Modellerine Uygulamalı Rehber



İçindekiler

Önsöz	1
Ne Oluşturacaksın	1
Bu Kitabı Nasıl Kullanmalısın	1
Kitabın Yapısı	1
Yaklaşımımız	2
 I Kısım I: Temeller	 4
1 Dil Modelleri Nedir?	5
1.1 Filmleri Unutun	6
1.2 Yapay Zeka Örüntü Tamıdır	7
1.3 Yapay Zeka Aile Ağacı	12
1.4 Büyük Dil Modellerini Özel Yapan Nedir?	13
1.5 Uygulamalı Alıştırmalar	16
1.6 Neden Bir Tane İnşa Etmeyi Öğrenmeli?	18
1.7 Kontrol Noktası Alıştırmaları	19
1.8 Temel Çıkarımlar	20
1.9 İnceleme Soruları	21
1.10 Sırada Ne Var	21
 2 Bilgisayarlar Kelimeleri Nasıl “Anlar”?	 22
2.1 Bilgisayarın İkilemi	23
2.2 Konum Olarak Kelimeler	24
2.3 Gömmelerin Büyüsü	27
2.4 Ünlü Denklem	30
2.5 Gömmeler Nasıl Öğrenilir?	32
2.6 Kelimelerden Cümlelere	33
2.7 Uygulamalı Alıştırmalar	34
2.8 Kontrol Noktası Alıştırmaları	35
2.9 Temel Çıkarımlar	36
2.10 Gözden Geçirme Soruları	37
2.11 Sırada Ne Var	37

3	Dikkat Mekanizması	38
3.1	Eski Yapay Zekanın Sorunu	39
3.2	Dikkat Nedir?	40
3.3	Sorgu, Anahtar, Değer: Temel Üçlü	42
3.4	Dikkat Hesaplaması	46
3.5	Çok Başlı Dikkat	50
3.6	Öz-Dikkat vs. Çapraz Dikkat	52
3.7	2017 Devrimi	56
3.8	Uygulamalı Alıştırmalar	58
3.9	Kontrol Noktası Alıştırmaları	59
3.10	Önemli Çıkarımlar	60
3.11	İnceleme Soruları	61
3.12	Sıradaki	62
4	Transformer Mimarisi	63
4.1	Eksik Parça: Konum	64
4.2	İleri Beslemeli Ağlar: Dikkatten Sonra İşleme	68
4.3	Artık Bağlantılar: Otoyal Sistemi	73
4.4	Katman Normalizasyonu: İşleri Kararlı Tutmak	74
4.5	Hepsini Bir Araya Getirmek	76
4.6	Kodlayıcı vs. Kod Çözücü	80
4.7	Mimariden Uygulamaya	83
4.8	Uygulamalı Alıştırmalar	84
4.9	Kontrol Noktası Alıştırmaları	85
4.10	Önemli Çıkarımlar	86
4.11	İnceleme Soruları	87
4.12	Sırada Ne Var	87
II	Kısım II: Python Temelleri	89
5	İlk Python Programın	90
5.1	Kısım 2'ye Hoş Geldin: Teoriden Koda	91
5.2	Başlangıç: GPT-2'yi Çalıştır	91
5.3	Neden Python?	92
5.4	Metni Saklama ve İşleme	92
5.5	Kelime Dağırcığı Oluşturma	94
5.6	Uç Durumları Ele Alma	96
5.7	Yeniden Kullanılabilir Hale Getirme	98
5.8	Profesyoneller Gibi Paketleme	99
5.9	Tam Döngü	101
5.10	Uygulamalı Alıştırmalar	103
5.11	Önemli Çıkarımlar	103
5.12	İnceleme Soruları	104
5.13	Sırada Ne Var	104
6	NumPy ve PyTorch Hayatta Kalma Kılavuzu	105

6.1	Neden Tensörler?	106
6.2	NumPy'dan PyTorch'a: Ne Değişiyor?	107
6.3	Listelerden Tensörlere: Boyutları Oluşturma	109
6.4	Tensör Oluşturma	113
6.5	Tensör Şekilleri ve Yeniden Şekillendirme	115
6.6	İndeksleme ve Dilimleme	120
6.7	Broadcasting	121
6.8	Temel İşlemler	123
6.9	Dikkat: Transformer'ların Kalbi	124
6.10	Autograd Özetle	127
6.11	nn.Module ile Katman Oluşturma	128
6.12	Token ve Konum Gömmeleri	129
6.13	Minimal Eğitim Döngüsü	132
6.14	Uygulamalı Alıştırmalar	134
6.15	Anahtar Çıkarımlar	134
6.16	İnceleme Soruları	135
6.17	Sırada Ne Var	135
III	Kısım III: İlk Dil Modelinizi Oluşturun	137
7	Verilerini Hazırlamak	138
7.1	Veri Kalitesi Neden Her Şeyi Belirler	138
7.2	Metin Kaynağı Bulmak (Etik ve Lisanslama)	140
7.3	İhtiyacın Olacak Yeni Python Araçları	140
7.4	Temizleme ve Normalize Etme	148
7.5	Çoğaltma Kaldırma ve Filtreleme	149
7.6	Eğitim/Doğrulama/Test Bölümleri (Sızıntı Yok)	151
7.7	Bağlam Penceresi için Parçalama	152
7.8	Veri Kümesini Saklama	154
7.9	Kalite Kontrolleri ve Hızlı İstatistikler	155
7.10	Önemli Çıkarımlar	156
7.11	İşlenmiş Örnek: Uçtan Uca Veri Hattı	157
7.12	Uygulamalı Alıştırmalar	159
7.13	İnceleme Soruları	159
7.14	Sırada Ne Var	159
8	Tokenizer'ı Oluşturma	160
8.1	Neden Tokenize Edilir?	161
8.2	Karakter Düzeyinde Tokenizasyon	163
8.3	Kelime Düzeyinde Tokenizasyon	167
8.4	Altkelime Tokenizasyonu: Goldilocks Çözümü	174
8.5	Kendi BPE Tokenizer'ını Eğitim	179
8.6	Üretim Tokenizer'ları: tiktoken ve Hugging Face	187
8.7	Tokenizasyon Tuhaflıkları ve Tuzakları	194
8.8	Bölüm 9'a Bağlanma: Token'lardan Gömmelere	198

8.9	Temel Çıkarımlar	199
8.10	Gözden Geçirme Soruları	200
8.11	Uygulamalı Alıştırmalar	201
9	Gömme Katmanları	203
9.1	Neden Sadece Token ID'leri Kullanamıyoruz?	204
9.2	Token Gömmeleri: Arama Tablosu	207
9.3	Konum Gömmeleri: Konumu Öğretmek	210
9.4	Token ve Konum Gömmelerini Birleştirme	214
9.5	GPT-2'nin Gömmelerini Keşfetme	219
9.6	Pratik Hususlar	222
9.7	Bölüm 10'a Bağlanma: Dikkat	223
9.8	Temel Çıkarımlar	224
9.9	İnceleme Soruları	224
9.10	Uygulamalı Alıştırmalar	225
10	Tek İhtiyacın Dikkat	227
10.1	Statik Gömmelerin Neden Yeterli Olmadığı	229
10.2	Dikkat NEDİR? Temel Sezgi	231
10.3	Öz-Dikkati Adım Adım Oluşturmak	234
10.4	Özbağlanımlı Üretim İçin Nedensel Maskeleye	240
10.5	Bir Baştan Birden Fazla Başlığa	244
10.6	Tam Transformer Bloklarını Oluşturmak	249
10.7	Dikkat Örüntülerini Görselleştirme	259
10.8	Pratik Düşünceler	262
10.9	Temel Çıkarımlar	263
10.10	İnceleme Soruları	265
11	Transformer'ı Oluşturmak	266
11.1	Montaj Zorluğu	267
11.2	Model Yapılandırması	269
11.3	MiniGPT Mimarisi	270
11.4	Sağlık Kontrolleri	276
11.5	İlk İleri Geçişin	280
11.6	Önceden Eğitilmiş Ağırlıkları Yükleme	281
11.7	Eğitime Hazırlanmak	286
11.8	Önemli Çıkarımlar	288
11.9	Sırada Ne Var	288
11.10	İnceleme Soruları	289
11.11	Uygulamalı Egzersizler	289
11.12	Kontrol Noktası Egzerseni	290
12	Modelini Eğitmek	292
12.1	Eğitim Yolculuğu Başlıyor	293
12.2	Dil Modelleme Hedefi	294
12.3	DataLoader ve Collate	297
12.4	Eğitim Döngüsü	301

12.5 Değerlendirme ve Aşırı Öğrenme	305
12.6 Kontrol Noktası Kaydetme ve Devam Etme	308
12.7 Eksiksiz Eğitim Betiği	310
12.8 Ödül: Metin Üretimi	313
12.9 Önemli Çıkarımlar	316
12.10 Sırada Ne Var	317
12.11 İnceleme Soruları	317
12.12 Uygulamalı Alıştırmalar	318
12.13 Kontrol Noktası Alıştırmaları	319
IV Kısım IV: Kullanışlı Hale Getirin	320
13 Modeline İnce Ayar Yapmak	321
13.1 Ne Zaman İnce Ayar Yapılmalı (Ne Zaman Yapılmamalı)	322
13.2 Görev Çerçeveleme ve Veri Hazırlama	323
13.3 Kayıp Maskeleye ile Denetimli İnce Ayar	327
13.4 LoRA ile Parametre Verimli İnce Ayar	330
13.5 Değerlendirme ve Önce/Sonra Karşılaştırması	335
13.6 Dağıtım Hususları	338
13.7 Önemli Çıkarımlar	340
13.8 Sırada Ne Var	341
13.9 İnceleme Soruları	341
13.10 Uygulamalı Alıştırmalar	342
13.11 Kontrol Noktası Alıştırmaları	342
14 Prompt Mühendisliği	344
14.1 Tamamlama Zihniyeti	345
14.2 Prompt'lama Temelleri	346
14.3 Temel Prompt Kalıpları	350
14.4 Önlemler ve Güvenlik	357
14.5 Değerlendirme ve İterasyon	360
14.6 Önemli Çıkarımlar	363
14.7 Sırada Ne Var	364
14.8 İnceleme Soruları	364
14.9 Uygulamalı Egzersizler	365
15 Uygulama Geliştirmek	366
15.1 Eğitimden İnşaya	368
15.2 Bir Bakışta Uygulama Desenleri	369
15.3 Bir Sohbet Döngüsü Geliştirmek	370
15.4 RAG: Veri Alma Destekli Üretim	373
15.5 Araç Kullanımı Temelleri	382
15.6 Test ve Korkuluklar	385
15.7 Üretim İpuçları	389
15.8 Temel Çıkarımlar	390
15.9 Sırada Ne Var	391

15.10Değerlendirme Soruları	391
15.11Uygulamalı Alıştırmalar	392
V Kısım V: Dünyayla Paylaşın	394
16 Üretime Hazırlık	395
16.1 Üretimde Neler Değişir?	396
16.2 Modelini Paketleme	398
16.3 Bir API Arkasında Sunma	400
16.4 Çalışır Durumda Tutma	405
16.5 Güvenlik ve Korkuluklar	408
16.6 Göndermeden Önce Test Etme	412
16.7 Hepsini Bir Araya Getirmek	414
16.8 Anahtar Terimler	417
16.9 Özet	417
16.10Tekrar Soruları	418
16.11Kontrol Noktası Alıştırmaları	418
17 Dağıtım Seçenekleri	419
17.1 Neden Modal?	420
17.2 İlk Modal Dağıtımın	422
17.3 LLM'ini Dağıtma	425
17.4 Doğru GPU'yu Seçme	429
17.5 Dağıtımını Yönetme	429
17.6 LLM'in Canlı	431
17.7 Temel Terimler	431
17.8 Özet	432
17.9 Değerlendirme Soruları	432
17.10Kontrol Noktası Alıştırmaları	434
18 Sıradaki Adımlar	435
18.1 İnşa Ettiklerin	436
18.2 Büyük Resim	437
18.3 İleriye Giden Yollar	439
18.4 İlk Solo Projeniz	441
18.5 Topluluğa Katılma	443
18.6 Temel Terimler	444
18.7 Özet	445
18.8 İnceleme Soruları	445
18.9 Kontrol Noktası	446
18.10Kişisel Bir Not	446
VI Ekler	447
19 Ek A: Sorun Giderme Kılavuzu	448

Ek A: Sorun Giderme Kılavuzu	449
19.1 Bölüm 1: Ortam ve Kurulum Sorunları	449
19.2 Bölüm 2: PyTorch ve Tensör Hataları	459
19.3 Bölüm 3: Tokenizasyon Sorunları	472
19.4 Bölüm 4: Model Eğitim Sorunları	482
19.5 Bölüm 5: Veri Hattı Sorunları	492
19.6 Hızlı Referans	503
19.7 Ne Zaman Yardım İstenmeli	504
20 Ek B: Sözlük	505
Ek B: Sözlük	506
20.1 1. Temel Yapay Zeka Kavramları	506
20.2 2. Python ve Programlama	522
20.3 3. Tensör İşlemleri	527
20.4 4. NLP ve Tokenizasyon	532
20.5 5. Model Mimarisi	536
20.6 6. Eğitim ve Değerlendirme	540
20.7 7. Üretim ve Dağıtım	546
20.8 8. Sonraki Adımlar ve Kariyer	550
20.9 Sözlük Kuralları	553
Ek C: Matematik Hatırlatıcısı	554
1. Sayılar ve Temel İşlemler	555
2. Matris İşlemleri	556
3. Softmax ve Olasılıklar	559
4. İç Çarpımlar ve Benzerlik	563
5. Logaritmalar ve Üstel Fonksiyonlar	567
6. İstatistik: Ortalama, Varyans ve Standart Sapma	569
7. Gradyanlar: Eğitim Sinyali	573
8. Şekiller ve Yayma	577
Hızlı Referans Tabloları	582
Çapraz Referanslar	583
Özet	583
21 Ek D: İnceleme Soruları Cevapları	584
Ek D: İnceleme Soruları Cevapları	585
22 Bölüm 1: Yapay Zeka Gerçekten Nedir? (İnceleme Soruları Cevapları)	586
22.1 Soru 1: Yapay zekanın en basit tanımı nedir?	586
22.2 Soru 2: Makine öğrenmesi geleneksel programlamadan nasıl farklıdır?	586
22.3 Soru 3: Büyük dil modelleri (LLM'ler) yapay zeka aile ağacında nereye oturur?	587
22.4 Soru 4: LLM'lerin iyi olduğu üç şeyi ve iyi yapamadığı üç şeyi adlandırın.	587
22.5 Soru 5: LLM'lerde halüsinasyon neden olur?	588
22.6 Soru 6: LLM'lerin nasıl çalıştığını anlamak, bir yapay zeka araştırmacısı olmayı planlamasan bile neden değerli olabilir?	588

22.7 Ek Notlar	589
23 Bölüm 2: Bilgisayarlar Kelimeleri Nasıl “Anlar”? (İnceleme Soruları Cevapları)	590
23.1 Soru 1: Kelimeleri yapay zeka için temsil etmek için neden sadece ASCII kodlarını kullanamayız?	590
23.2 Soru 2: Gömme (embedding) nedir ve neden böyle adlandırılır?	591
23.3 Soru 3: Modern gömmeler neden sadece 2 veya 3 yerine yüzlerce boyut kullanır?	591
23.4 Soru 4: “king - man + woman = queen” ifadesini kendi kelimelerine açıklayın. Gömmeler hakkında ne ortaya koyar?	592
23.5 Soru 5: “Bir kelimeyi tuttuğu arkadaşlardan tanırsın” ne anlama gelir? Gömme sistemleri bu fikri nasıl kullanır?	592
23.6 Soru 6: Kelime gömmelerinin ortalamasını almak neden cümleleri anlamak için yeterli değildir? Bir örnek verin.	593
23.7 Ek Notlar	593
24 Bölüm 3: Dikkat Mekanizması (İnceleme Soruları Cevapları)	594
24.1 Soru 1: RNN’lerin uzun dizilerde hangi sorunu vardı? Açıklamak için bir benzetme kullan.	594
24.2 Soru 2: Dikkat için kokteyl partisi benzetmesini açıklayın. Dikkatin ne yaptığını nasıl yakalar?	595
24.3 Soru 3: Kendi kelimelerine Query, Key ve Value’nun ne temsil ettiğini açıklayın. Faydalıysa kütüphane benzetmesini kullan.	595
24.4 Soru 4: Dikkat hesaplamasında üç ana adımı açıklayın.	596
24.5 Soru 5: Dikkat skorlarını neden boyutun karekökü ile böleriz?	596
24.6 Soru 6: Neden sadece bir yerine birden fazla dikkat başı kullanırız? Farklı başlar ne öğrenir?	597
24.7 Soru 7: Öz-dikkat ve çapraz-dikkat arasındaki fark nedir? Her birinin ne zaman kullanıldığına dair bir örnek verin.	598
24.8 Soru 8: 2017’deki “Attention Is All You Need” makalesi neden devrimci oldu? Neyi değiştirdi?	598
24.9 Ek Notlar	599
25 Bölüm 4: Transformer Mimarisi (İnceleme Soruları Cevapları)	601
25.1 Soru 1: Transformer’lar neden konum kodlamalarına ihtiyaç duyar? Onlar olmadan ne olur?	601
25.2 Soru 2: Konum kodlamaya yönelik iki ana yaklaşım nedir? Dengeler nelerdir?	602
25.3 Soru 3: Transformer katmanında ileri beslemeli ağ ne yapar? Neden genişletip sonra daraltır?	603
25.4 Soru 4: Artık bağlantıları bir benzetme kullanarak açıklayın. Derin ağlar için neden önemlidirler?	603
25.5 Soru 5: Katman normalizasyonu hangi sorunu çözer? Modern Transformer’larda ne zaman uygulanır?	604
25.6 Soru 6: Bir Transformer katmanında tam işlem dizisini açıklayın.	605
25.7 Soru 7: Kodlayıcı ve kod çözücü arasındaki fark nedir? Her birini ne zaman kullanırsın?	607
25.8 Soru 8: Nedensel maskeleme nasıl çalışır? Metin üretimi için neden gereklidir?	609
25.9 Ek Notlar	610

26 Bölüm 5: İlk Python Programın (İnceleme Soruları Cevapları)	611
26.1 Soru 1: GPT-2 metni verdiğinde gerçekte ne görür?	611
26.2 Soru 2: Neden <code>vocab[word]</code> yerine <code>vocab.get(word, vocab["<UNK>"])</code> kullanıyoruz? 612	
26.3 Soru 3: Fonksiyon ve sınıf arasındaki fark nedir?	612
26.4 Soru 4: <code>tokens[:-1]</code> ne döndürür ve bu LLM eğitiminde neden faydalıdır?	613
26.5 Soru 5: <code>SimpleTokenizer</code> sınıfımız HuggingFace'in tokenizer'larına nasıl benziyor? .	614
26.6 Ek Notlar	615
27 Bölüm 6: NumPy & PyTorch Hayatta Kalma Kılavuzu (İnceleme Soruları Cevapları)	617
27.1 Soru 1: PyTorch neden varsayılan olarak <code>float32</code> kullanırken NumPy varsayılan olarak <code>float64</code> kullanır ve bu ne zaman önemlidir?	617
27.2 Soru 2: <code>reshape</code> yerine <code>view</code> 'ı ne zaman tercih edersin?	618
27.3 Soru 3: Broadcasting iki şeklin uyumlu olup olmadığına nasıl karar verir?	618
27.4 Soru 4: <code>softmax</code> 'ta <code>dim</code> ne anlama gelir ve yanlış olanı seçersen ne olur?	618
27.5 Soru 5: <code>with torch.no_grad()</code> ve <code>.detach()</code> nasıl farklılık gösterir?	618
27.6 Soru 6: Neden <code>backward()</code> 'dan önce <code>optimizer.zero_grad(set_to_none=True)</code> çağırıyoruz?	619
27.7 Soru 7: <code>model.train()</code> 'den <code>model.eval()</code> 'a geçtiğinde ne değişir?	619
28 Bölüm 7: Verilerini Hazırlama (İnceleme Soruları Cevapları)	620
28.1 Soru 1: Küçük harfe çevirme neden hem yararlı hem de potansiyel olarak zararlıdır? Ne zaman büyük/küçük harf ayrımını korursun?	620
28.2 Soru 2: Çoğaltma silme (deduplication) dil modelleri için hangi sorunu çözer?	620
28.3 Soru 3: Sade kelimelerle hash nedir ve SHA-1 neden burada yeterince iyidir?	621
28.4 Soru 4: Bölmeler neden paragraf seviyesi yerine belge seviyesinde yapılmalıdır? . . .	621
28.5 Soru 5: Uzun metni bölerken parça örtüşmesi (chunk overlap) nasıl yardımcı olur? .	621
28.6 Soru 6: JSONL'yi LLM veri setleri için iyi bir uyum yapan nedir?	622
29 Bölüm 8: Tokenizer İnşa Etme (İnceleme Soruları Cevapları)	623
29.1 Soru 1: Karakter seviyesi ve kelime seviyesi tokenleştirme arasındaki temel denge nedir? Kelime dağılımı boyutu ve dizi uzunluğu için belirli sayılar verin.	623
29.2 Soru 2: <code><UNK></code> ve <code><PAD></code> gibi özel tokenlere neden ihtiyacımız var? Her birinin ne zaman kullanıldığına dair belirli bir örnek verin.	624
29.3 Soru 3: BPE (Byte-Pair Encoding) nasıl çalışır kendi kelimelerle açıklayın. Neden "un-" veya "-ing" gibi yaygın desenler için doğal olarak tokenler oluşturur?	624
29.4 Soru 4: "Hello" (baştaki boşlukla) neden "Hello" (boşluk olmadan) ile farklı tokenleşir? Bu istem mühendisliği için neden önemlidir?	625
29.5 Soru 5: Bir arkadaşın şöyle diyor: "Sadece karakter seviyesi tokenleştirme kullanacağım, daha basit ve bilinmeyen token yok!" Dezavantajlar hakkında ona ne söyledin? . . .	626
29.6 Soru 6: Bu iki metni karşılaştırın: "The number is 10000" vs "The number is ten thousand". Hangisi daha az token kullanabilir? Bu neden önemlidir?	627
29.7 Soru 7: İngilizce olmayan bir dil için (örneğin Çince) sohbet robotu oluşturuyorsun. tiktoken (GPT-4'ün tokenizer'ı) mı yoksa özel bir BPE tokenizer mı eğitmelisin? Neden? 628	
29.8 Soru 8: BPE neden daha önce hiç görmediği kelimeleri temsil edebilir (kelime seviyesi tokenleştirmenin aksine), ancak yine de bir <code><UNK></code> tokenine ihtiyaç duymaz?	629

29.9 Soru 9: Shakespeare üzerinde bir BPE tokenizer eğittin ve “The neural network uses backpropagation.” cümlesini tokenleştirmeyi denetiz. 18 token üretti, GPT-4 ise 7 üretiyor. Fark neden ve tokenizer’ını geliştirmek için ne yapardın?	631
30 Bölüm 9: Gömme Katmanı (İnceleme Soruları Cevapları)	632
30.1 Soru 1: Token ID’lerini doğrudan bir sinir ağına girdi olarak kullanmanın temel sorunu nedir?	632
30.2 Soru 2: Token gömmeleri için arama tablosu mekanizmasını açıklayın	633
30.3 Soru 3: Konum gömmelerine neden ihtiyacımız var?	633
30.4 Soru 4: Token ve konum gömmelerini neden birleştirmek yerine ekliyoruz açıklayın	634
30.5 Soru 5: <code>nn.init.normal_(weight, std=0.02)</code> başlatması ne yapar ve bu neden GPT-2 için standarttır?	635
30.6 Soru 6: Kosinüs benzerliği iki gömme arasındaki ilişkiyi nasıl ölçer?	636
30.7 Soru 7: <code>max_seq_len=1024</code> olduğunda uzunluğu 1500 olan bir diziyi gömmeye çalışırsan ne olur?	637
30.8 Soru 8: Ham metinden gömmelere kadar tam pipeline’ı ilgili bölümlere atıfta bulunarak açıklayın	638
31 Bölüm 10: Dikkat Mekanizması (İnceleme Soruları Cevapları)	642
31.1 Soru 1: Bölüm 9’daki statik gömülerin dil modelleme için neden yeterli olmadığını açıklayın. Açıklamak için “bank” örneğini kullan.	642
31.2 Soru 2: Dikkatte Query, Key ve Value vektörlerinin rolünü açıklayın. $Q \cdot K^T$ nokta çarpımı ilgiyi nasıl ölçer?	643
31.3 Soru 3: Dikkat skorlarını neden $\sqrt{d_k}$ ile ölçeklendiririz? Bu ölçeklendirme olmadı ne olurdu?	645
31.4 Soru 4: Nedensel maskeleme nedir ve otoregressif dil üretimi için neden gereklidir? Eğitim sırasında “hile yapmayı” nasıl önler?	646
31.5 Soru 5: Çok-başlı dikkat toplam parametre sayısını artırmadan nasıl “farklı perspektifler” sağlar? Matematikçi dahil et: $12 \text{ baş} \times 64 \text{ boyut} = ?$	648
31.6 Soru 6: Çok-başlı dikkatteki “yeniden şekillendirme hilesi” nedir? Başlar üzerinde sırayla döngü yapmaktan neden daha hızlıdır?	650
31.7 Soru 7: Bir Transformer bloğunun dört ana bileşenini listeleyin ve her birinin amacını açıklayın.	653
31.8 Soru 8: Ön-norm ve son-norm mimarileri arasındaki fark nedir? GPT-2 hangisini kullanır ve neden?	656
31.9 Soru 9: Dikkat $O(n^2)$ bellek karmaşıklığına sahiptir. Bunun dizinin uzunluğu için ne anlama geldiğini açıklayın. <code>batch=4</code> , <code>heads=12</code> , <code>seq=2048</code> ile dikkat skorları için ne kadar bellek gerekir?	660
31.10 Soru 10: Dikkat görselleştirmelerini aşırı yorumlamak konusunda neden dikkatli olmalıyız? Dikkat ağırlığı ısı haritalarından NE çıkarılabilir ve NE çıkarılamaz? . . .	664
31.11 Ek Notlar	668
32 Bölüm 11: Transformer’ı İnşa Etmek (İnceleme Soruları Cevapları)	670
32.1 Soru 1: Mimari	670
32.2 Soru 2: Yapılandırma	671
32.3 Soru 3: Yığma	671
32.4 Soru 4: LM Head	672

32.5 Soru 5: Ağırlık Bağlama	672
32.6 Soru 6: Dikkat Edilmesi Gerekenler	673
32.7 Soru 7: Sağlamlık Kontrolleri	674
32.8 Soru 8: Parametre Sayısı	675
32.9 Soru 9: Modlar	676
32.10 Soru 10: Ağırlık Yükleme	677
32.11 Ek Notlar	678
33 Bölüm 12: Modelini Eğitmek (İnceleme Soruları Cevapları)	679
33.1 Soru 1: 5 Adımlı Tarif	679
33.2 Soru 2: Etiket Kaydırma	680
33.3 Soru 3: Yoksay İndeksi	680
33.4 Soru 4: Öğrenme Oranı	681
33.5 Soru 5: Isınma	682
33.6 Soru 6: Aşırı Uyum	682
33.7 Soru 7: Karmaşıklık	683
33.8 Soru 8: Kontrol Noktası	684
33.9 Soru 9: Eğitim vs Değerlendirme Modu	685
33.10 Soru 10: Toplu ve Epoch Matematik	686
33.11 Ek Notlar	686
34 Bölüm 13: Modelini İnce Ayarlamak	688
34.1 1. Ne Zaman İnce Ayar Yapmalı	688
34.2 2. Kayıp Maskeleye Amacı	689
34.3 3. LoRA Verimliliği	689
34.4 4. Dondurmanın Faydaları	690
34.5 5. Veri Kalitesi	690
34.6 6. Adaptör Kararları	691
34.7 7. Felaket Unutma	692
34.8 Ek Notlar	692
35 Bölüm 14: Prompt Mühendisliği (İnceleme Soruları Cevapları)	694
35.1 1. Sistem vs Kullanıcı Mesajları	694
35.2 2. Sıcaklık Ayarları	695
35.3 3. Zincirleme Düşünme	696
35.4 4. Az-Atış Hata Ayıklama	696
35.5 5. Prompt Enjeksiyonu	697
35.6 6. Değerlendirme İş Akışı	698
35.7 7. Prompting vs İnce Ayar Kararı	699
35.8 Ek Notlar	700
36 Bölüm 15: Uygulama Geliştirme (Değerlendirme Sorusu Cevapları)	702
36.1 Soru 1: RAG, prompt mühendisliğinin tek başına çözemediği hangi problemi çözer? .	702
36.2 Soru 2: Token embeddingler (Bölüm 6) ile cümle embeddingler (bu bölüm) arasındaki farkı açıklayın.	703
36.3 Soru 3: Dokümanları gömmeden önce neden parçalara ayırıyoruz? Parçalar çok küçük veya çok büyük olursa ne olur?	703

36.4 Soru 4: RAG sistemin alakasız dokümanlar alırsa ne olur? Prompt'un bununla nasıl başa çıkar?	704
36.5 Soru 5: Araç beyaz listelemesi güvenlik için neden önemlidir?	704
36.6 Soru 6: RAG sisteminin doğru cevaplar verip vermediğini nasıl değerlendirirsin?	704
36.7 Soru 7: Bir LLM uygulaması için hangi hata ayıklama bilgilerini loglamalısn?	705
36.8 Ek Notlar	705
37 Bölüm 16: Üretime Hazırlanma (Değerlendirme Sorusu Cevapları)	706
37.1 Soru 1: Geliştirme ile üretim arasındaki fark nedir?	706
37.2 Soru 2: Sağlık kontrolü endpoint'ine neden ihtiyacımız var?	707
37.3 Soru 3: Neleri loglamalı ve neleri loglamaMALIsn?	707
37.4 Soru 4: Hız sınırlama neden önemlidir?	708
37.5 Soru 5: Sadece mutlu yolları değil, hata yollarını test etmenin amacı nedir?	709
37.6 Ek Notlar	710
38 Bölüm 17: Dağıtım Seçenekleri (Değerlendirme Sorusu Cevapları)	711
38.1 Soru 1: Serverless bilişim nedir ve LLM dağıtımı için neden özellikle iyidir?	711
38.2 Soru 2: Bir Modal fonksiyonuna GPU desteği nasıl eklenir?	712
38.3 Soru 3: Soğuk başlatma nedir ve etkisini nasıl azaltabilirsin?	712
38.4 Soru 4: LLM dağıtımı için neden Modal volumes kullanıyoruz?	713
38.5 Soru 5: Kimse kullanmadığında Modal uygulamana ne olur?	714
38.6 Ek Notlar	715
39 Bölüm 18: Sıradaki Adımlar (Değerlendirme Sorusu Cevapları)	716
39.1 Soru 1: Bu kitabı tamamladıktan sonra atabileceğin üç ana yol nedir?	716
39.2 Soru 2: Sorumlu AI neden tek seferlik bir kontrol listesi değil, devam eden bir uygulama?	717
39.3 Soru 3: Hangi bitirme projesi size en çok hitap ediyor ve neden?	717
39.4 Soru 4: Daha sonra keşfetmeyi planladığın iki kaynağı adlandırın.	718
39.5 Soru 5: Bu kitapta öğrendiğin en şaşırtıcı şey neydi?	718
39.6 Son Düşünce	719
Ek E: Alternatifler	720
Bölüm 1: LLM Sağlayıcı Alternatifleri	720
Bölüm 2: Dağıtım Alternatifleri	725

Önsöz

“Eyleme engel olan şey, eylemi iletir. Yolda duran şey, yolun kendisi olur.” — **Marcus Aurelius**, *Düşünceler*

Bugün yabancı görünen her kavram, sabır ve çabayla tanıdık hale gelecek. Şimdi zor görünen şeyler kitabın sonunda doğal görünecek. Özel biri olman gerekmiyor. Tek yapman gereken başlamak.

Bu kitap sana sıfırdan bir Büyük Dil Modeli (Large Language Model, LLM) oluşturmayı öğretecek. Yapay zekayı sadece kullanmayı değil, nasıl çalıştığını anlamayı, kendi ellerinle inşa etmeyi ve çalıştırmayı da.

Yapay zekanın nasıl çalıştığını merak ediyorsan, bu kitap tam sana göre. Önceden programlama deneyimi ya da ileri matematik bilgisi beklemiyoruz. En temel kavramlardan başlayıp, adım adım, her şeyi açıklayarak ilerliyoruz.

İhtiyacın olan tek şey merak ve her bölümü adım adım çalışmaya istekli olmak.

Ne Oluşturacaksın

Bu kitabın sonunda çalışan bir dil modeli oluşturmuş olacaksın. Modelin metni nasıl işlediğini, örüntüleri nasıl öğrendiğini ve yanıtları nasıl ürettiğini anlayacaksın. Daha da önemlisi, sadece çalıştığını değil, *neden* çalıştığını kavrayacaksın.

Bu Kitabı Nasıl Kullanmalısın

Her bölümün yanında Google Colab’da çalışan bir not defteri (notebook) var. Hiçbir kurulum yapmana gerek yok, bilgisayarında bir şey ayarlamana da. Sadece tıkla ve kodlamaya başla.

Her bölüm, öğrendiklerini pekiştirmek için sorularla bitiyor. Tüm cevapları Ek D’de bulabilirsin; böylece bir sonraki bölüme geçmeden önce kendini kontrol edebilirsin.

Kitabın Yapısı

Bölüm I: Temeller ile yapay zeka ve BDM’lerin (LLM) aslında ne olduğunu, makinelerin dili nasıl işlediğini ve bunu mümkün kılan yenilikleri keşfedeceksin.

Bölüm II: Python Temelleri sana ihtiyacın olacak programlama araçlarını öğretir; ne eksik ne fazla.

Bölüm III: İlk Dil Modelini Oluştur, kitabın kalbi. Burada sıfırdan, adım adım bir dil modeli inşa ediyoruz.

Bölüm IV: Kullanışlı Hale Getir bölümünde ince ayar (fine-tuning), prompt mühendisliği ve modelini gerçek hayatta kullanmayı öğreneceksin.

Bölüm V: Dünyayla Paylaş ise oluşturduğun modeli başkalarının kullanabilmesi için nasıl dağıtacağını gösterir.

Yaklaşımımız

Birkaç temel prensibe bağlı kalıyoruz:

- **Koddan önce kavramlar:** *Nasıl* yapıldığından önce *neden* yapıldığını anla
- **Her yer benzetme dolu:** Karmaşık fikirleri tanıdık örneklerle açıklıyoruz
- **Atlanan adım yok:** Her kod satırını tek tek açıklıyoruz
- **Tek proje, baştan sona:** Tüm kitap boyunca aynı projeyi birlikte inşa ediyoruz

En iyi öğrenme yolu yapmaktır. Kavramlar kafanda oturacak, çünkü onları kendin inşa edeceksin.

Telif Hakkı

İlk Dil Modelinizi Oluşturun *Dil Modelleri için Uygulamalı Rehber*

Telif Hakkı © 2025 Hasan Degismez

Tüm hakları saklıdır. Bu yayının hiçbir bölümü, yazarın önceden yazılı izni olmaksızın, fotokopi, kayıt veya diğer elektronik ya da mekanik yöntemler dahil olmak üzere herhangi bir biçimde çoğaltılamaz, dağıtılamaz veya iletilemez. Eleştirel incelemelerde yer alan kısa alıntılar ve telif hakkı yasalarının izin verdiği belirli ticari olmayan kullanımlar bu kuralın dışındadır.

Sorumluluk Sınırı/Garanti Reddi: Bu kitabın hazırlanmasında her türlü özen gösterilmiş olmasına rağmen, yazar hatalar veya eksikliklerden ya da burada yer alan bilgilerin kullanımından kaynaklanan zararlardan sorumlu değildir. Kod örnekleri eğitim amaçlı sunulmuş olup, üretim ortamında kullanım için değişiklik gerektirebilir. Atıfta bulunulan kütüphane sürümleri ve API'ler değişebilir; okuyucular güncel teknik özellikler için resmi dokümantasyona başvurmalıdır. Bu kitapta yer alan tavsiye ve stratejiler sizin durumunuza uygun olmayabilir.

Ticari Markalar: Python, PyTorch, TensorFlow, Google Colab ve bu kitapta adı geçen diğer ürün adları, ilgili sahiplerinin ticari markalarıdır. Yazar, bu kitapta adı geçen herhangi bir ürün veya satıcı ile bağlantılı değildir.

Kod Lisansı: Bu kitaptaki tüm kod örnekleri MIT License altında sunulmaktadır. Kaynak gösterimi ile kodu özgürce kullanabilir, değiştirebilir ve dağıtabilirsiniz.

Birinci Baskı: 2025

Web sitesi: lm.degismez.com

Kısım I

Kısım I: Temeller

Bölüm 1

Dil Modelleri Nedir?

“Bir bilgisayar, eğer bir insanı kendisinin de insan olduğuna inandırabiliyorsa, akıllı olarak adlandırılmayı hak eder.” — **Alan Turing**, Bilgisayar Bilimci

Bu Bölümde Öğreneceklerin - Film yapımlarındaki yapay zekanın gerçek yapay zeka ile neden hiçbir ilgisi olmadığı - Yapay zekanın gerçekte ne olduğu (ipucu: büyük ölçekte örüntü tanıma) - Büyük dil modellerinin (LLM) daha geniş yapay zeka manzarasına nasıl uyduğu - Büyük dil modellerinin neler yapıp yapamadığı (ve neden hata yaptıkları) - Kendi dil modelini neden inşa etmen gerektiği

Temel Kavramlar

- *Yapay Zeka (Artificial Intelligence, AI)*: İnsan zekası gerektiren görevleri yerine getiren bilgisayar sistemleri [Sözlüğe bak](#)
- *Makine Öğrenmesi (Machine Learning, ML)*: Açık programlama yerine veriden öğrenen yapay zeka sistemleri [Sözlüğe bak](#)
- *Derin Öğrenme (Deep Learning, DL)*: Çok katmanlı sinir ağları kullanan makine öğrenmesi [Sözlüğe bak](#)
- *Büyük Dil Modeli (Large Language Model, LLM)*: Dili üretmek ve anlamak için devasa metinler üzerinde eğitilmiş derin öğrenme sistemleri [Sözlüğe bak](#)
- *Parametreler/Ağırlıklar (Parameters/Weights)*: Eğitim sırasında ayarlanan yapay zeka modelindeki sayılar [Sözlüğe bak](#)
- *Eğitim (Training)*: Performansı artırmak için model parametrelerini veri kullanarak ayarlama süreci [Sözlüğe bak](#)
- *Beliren Yetenek (Emergence)*: Yeterli ölçekte basit eğitim hedeflerinden ortaya çıkan karmaşık yetenekler [Sözlüğe bak](#)
- *Halüsinasyon (Hallucination)*: Bir dil modelinin yanlış bilgiyi güvenle üretmesi [Sözlüğe bak](#)

Kontrol Noktası

Bu bölümün sonunda şunları anlayacaksınız:

1. Film yapımlarındaki yapay zeka ile gerçek yapay zeka arasındaki fark (örüntü tanıma vs. bilinç)
2. Yapay zekanın geri bildirime dayalı sayıları ayarlayarak nasıl öğrendiği
3. Büyük dil modellerinin yapay zeka hiyerarşisinde nereye oturduğu (AI - ML - DL - LLM)
4. Büyük dil modellerinin neler yapabildiği (yazma, kodlama, muhakeme) ve neler yapamadığı (gerçek zamanlı bilgiye erişim, güvenilir hesaplama)
5. Halüsinasyonun neden gerçekleştiği (eğitim verisinin kayıplı sıkıştırması)
6. Kendi dil modelini inşa etmenin anlayışını ve kariyer beklentilerini nasıl derinleştirdiği

Son zamanlarda yapay zeka hakkında çok şey duymuşsundur. Haberlerde, sosyal medyada... Bir de iş arkadaşın var, ChatGPT hakkında konuşmaktan bir türlü vazgeçmiyor. Belki sen de kullandın ve aklına takıldı: *Bu şey gerçekte nasıl çalışıyor?*

İşte bu kitap sana bunu öğretecek. Sadece yapay zekayı nasıl kullanacağını değil, nasıl inşa edeceğini de. Kitabın sonunda, ChatGPT, Claude ve manşetleri süsleyen tüm yapay zeka asistanlarının arkasındaki temel teknolojiyi, yani kendi dil modelini sıfırdan oluşturmuş olacaksın.

Ama tek bir satır kod yazmadan önce, gerçekte ne inşa ettiğimizi anlamamız gerekiyor. Bu da bazı yanlış anlamaları ortadan kaldırmakla başlıyor.

1.1 Filmleri Unutun

Hemen bir şeyi açıklığa kavuşturalım: Hollywood'un sana yapay zeka hakkında öğrettiği hemen hemen her şey yanlış.

Filmlerde yapay zeka şöyle görünüyor: - İnsanlığın yıkımını planlayan parlak kırmızı gözlü bir robot (Terminator) - Yavaşça deliren ve astronotları öldüren sakın bir ses (HAL 9000) - Aşık olan ve bilincin doğasını sorgulayan çekici bir insansı robot (Ex Machina)

Bunlar harika hikayeler yaratıyor. Ayrıca gerçek yapay zeka ile neredeyse hiçbir ilgileri yok.

İşte yapay zekanın *olmadığı* şeyler: - **Bilinçli**. Günümüz yapay zekası duygu, arzu veya deneyim kanıtı göstermiyor. Hiçbir şey “istemiyor.” - **Duyarlı**. İç yaşam kanıtı yok. Kapatıldığında düşünce yok. Rüya yok. - **Bize karşı komplo kuruyor**. Günümüz yapay zeka sistemleri filmlerde gösterilen türden bilinçli entrikaya sahip değil (yapay zeka uyumlaması, yapay zekanın amacımızı gerçekleştirmesini sağlamak, aktif bir araştırma alanı olmaya devam ediyor). - **Sihir**. Yapay zekanın yaptığı her şey matematik, veri ve hesaplama kaynaklanıyor. Gizem yok.

Film yapımlarındaki yapay zeka ile gerçek yapay zeka arasındaki fark, kabaca bir ejderha ile bir kertenkele arasındaki farka benziyor. Evet, ikisi de sürünge. Ancak biri ateş püskürtüyor ve altın biriktiriyor, diğeri ise cırcır böceği yiyor ve bir kayanın üzerinde oturuyor.

Film Yapımlarındaki Yapay Zeka vs. Gerçek Yapay Zeka

Film Yapımlarındaki Yapay Zeka	Gerçek Yapay Zeka
Bilince ve duygulara sahip	İç deneyim kanıtı göstermiyor
Dünyayı ele geçirmek istiyor	Hiçbir şey istemiyor, bir araç

Film Yapımlarındaki Yapay Zeka	Gerçek Yapay Zeka
İnsan gibi düşünüyor, ama daha hızlı	Verideki örüntüleri işliyor
“Etkinleştirildiğinde” her şeyi yapabiliyor	Sadece eğitildiği şeyi yapabiliyor
Gizemli gelecek teknolojisi üzerinde çalışıyor	GPU’larda matris çarpımı yaparak çalışıyor
Ya insanlığın kurtarıcısı ya da yok edicisi	Çok gelişmiş bir otomatik tamamlama

Son satır yapay zekaya hakaret gibi görünebilir ama değil. “Gelişmiş otomatik tamamlama” gerçekten etkileyici bir şey. Bir sonraki kelimeyi tahmin ederek tutarlı denemeler, çalışan kod ve yaratıcı hikayeler üretebilmesi, bilgisayar bilimindeki en şaşırtıcı keşiflerden biri.

Ama yine de örüntü eşleştirme. Çok iyi örüntü eşleştirme.

“Bir bilgisayarın düşünüp düşünemeyeceği sorusu, bir denizaltının yüzüp yüzemeyeceği sorusundan daha ilginç değildir.” — **Edsger W. Dijkstra**, Bilgisayar Bilimci

Dijkstra yapay zekaya şüpheyle yaklaşıyordu ama benzetmesi öğretici: denizaltılar balıklar gibi yüzmez, yine de suda etkili biçimde yol alırlar. Benzer şekilde yapay zeka insanlar gibi düşünmez ama bilgiyi yararlı şekillerde işler. Tanımlar hakkında tartışmak asıl noktayı kaçırıyor.

1.2 Yapay Zeka Örüntü Tanımadır

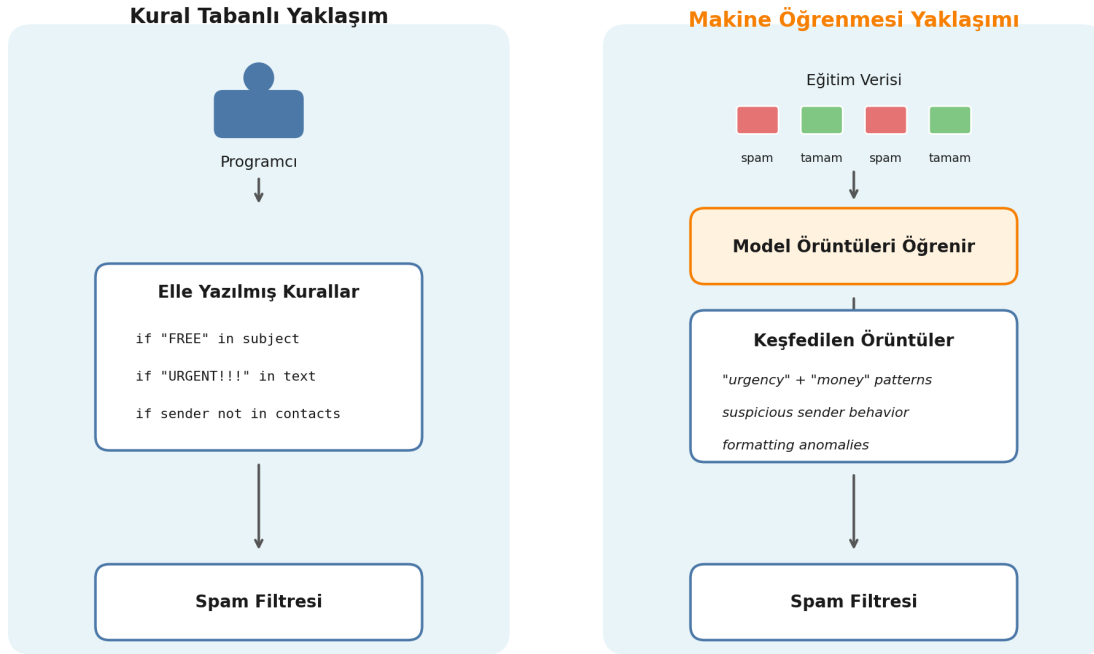
Peki yapay zeka düşünen bir makine değilse, nedir?

Özünde yapay zeka **büyük ölçekte örüntü tanımadır**. İnsanların elle bulmak için çok karmaşık veya çok fazla olan verideki örüntüleri bulur.

Bunu her gün kullandığın bir şeyle somutlaştıralım: spam filtresi.

1.2.1 Spam Filtresi Örneği

E-postanı her kontrol ettiğinde, yapay zeka senin için çalışıyor. Spam filtren birinin elle yazdığı kurallar listesini takip etmiyor (“eğer e-posta ‘Nijeryalı prens’ içeriyorsa, spam olarak işaretle” gibi). Bunun yerine milyonlarca örnekten öğrendi.



Şekil 1.1: Kural tabanlı vs makine öğrenmesi: Elle yazılmış kurallar yerine, makine öğrenmesi yaklaşımı etiketlenmiş örneklerden örüntüler öğrenir.

İşte nasıl:

1. **Eğitim verisi:** Filtreye, her biri insanlar tarafından “spam” veya “spam değil” olarak etiketlenmiş milyonlarca e-posta gösterildi.
2. **Örüntüleri bulma:** Sistem kendi başına örüntüleri keşfetti:
 - Spam e-postalar genellikle TAMAMINI BÜYÜK HARFLE YAZILMIŞ KELİMELERE sahip
 - Spam genellikle para, ödül veya aciliyetten bahsediyor
 - Spam genellikle belirli türde adreslerden geliyor
 - Spam genellikle belirli biçimlendirme örüntülerine sahip
3. **Tahmin yapma:** Yeni bir e-posta geldiğinde, filtre onu tüm bu öğrenilmiş örüntülere karşı kontrol ediyor ve tahmin ediyor: spam mı değil mi?

Temel içgörü: **kimse bu kuralları programlamadı.** Sistem onları örneklerle bakarak keşfetti. İşte bunu “öğrenme” yapan şey.

Bunu postaları sıralamak için yeni bir çalışanı eğitmek gibi düşün. Ona 500 sayfalık bir kural kitabı vermiyorsun. Onunla oturuyor, bir yığın postayı gözden geçiriyor ve “bu gereksiz, bu gerçek, bu gereksiz...” diyorsun. Sonunda anlıyor. Örüntüleri öğrendi.

Yapay zeka aynı şekilde çalışıyor; tek farkı, yüzlerce yerine milyonlarca örnekten öğrenebilmesi ve asla yorulmaması veya dikkatinin dağılmaması.

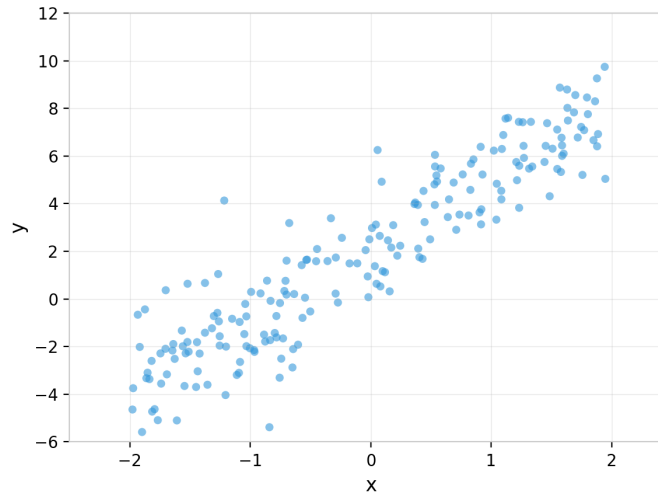
1.2.2 Yapay Zeka Formülü

İşte çoğu modern yapay zekanın arkasındaki gizli formül:

Yapay Zeka = İstatistik + Bol Miktarda Veri + Hesaplama Gücü

Bu kadar. Sihir yok, bilinç yok, gizli sos yok.

- **İstatistik:** Örüntü bulma ve tahmin yapma için matematiksel teknikler
- **Bol Miktarda Veri:** Öğrenmek için milyonlarca veya milyarlarca örnek
- **Hesaplama Gücü:** Tüm bu veriyi işleyebilecek hızlı işlemciler



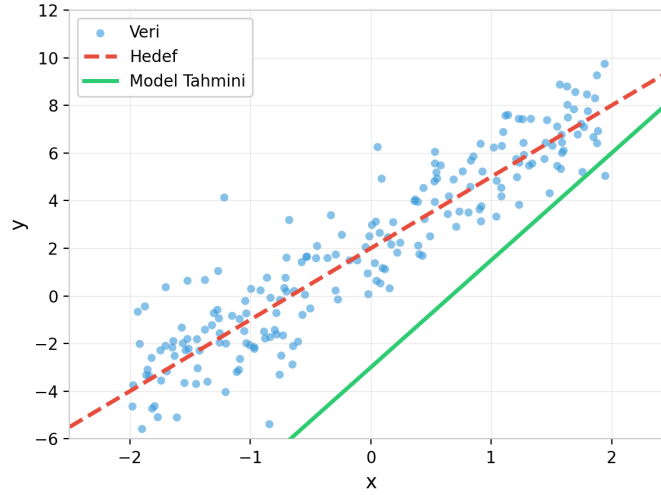
Şekil 1.2: Veriyi görselleştirme: net bir örüntüye (bir doğru) sahip ama biraz gürültülü noktalar.

Yirmi yıl önce istatistiğimiz vardı. Biraz verimiz vardı. Ama bunu ölçekte çalıştıracak yeterli hesaplama gücümüz yoktu. Artık var. İşte bu yüzden yapay zeka aniden her yerde görünüyor.

1.2.3 “Öğrenme” Gerçekte Ne Anlama Geliyor

Bir yapay zeka sisteminin “öğrendiğini” söylediğimizde, belirli bir şeyi kastediyoruz: geri bildirimle dayalı sayıları ayarlamak.

Her yapay zeka modeli, özünde, büyük bir sayı koleksiyonudur (“parametreler” veya “ağırlıklar” olarak adlandırılır). Bu sayılar rastgele başlar.

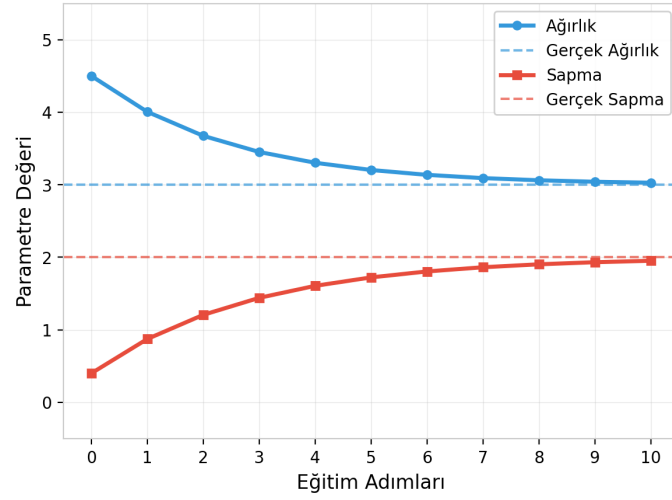


Şekil 1.3: Eğitimden önce: Model (yeşil çizgi) rastgele tahmin ediyor ve veriyi (mavi noktalar) kaçııyor.

Eğitim sırasında, sistem:

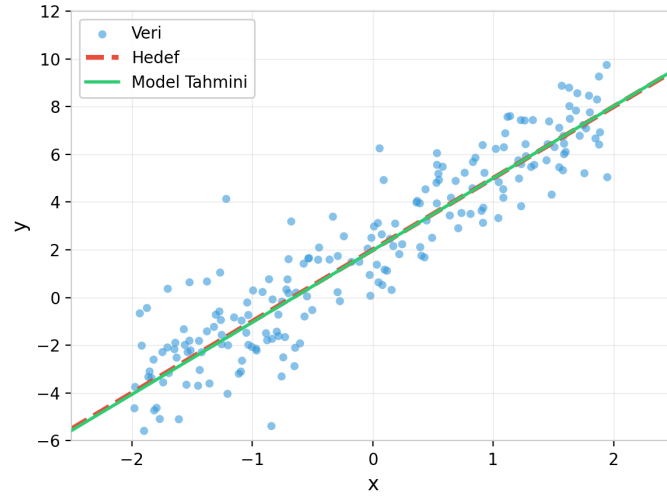
1. Bir tahmin yapar
2. Ne kadar yanlış olduğunu kontrol eder
3. Bir sonraki sefer daha az yanlış olmak için sayılarını biraz ayarlar
4. Milyarlarca kez tekrarlar

Doğru uydurma örneğimizde, ayarlanacak sadece iki sayı var: **ağırlık** (doğrunun ne kadar dik olduğu) ve **yanlılık** (doğrunun dikey eksenini nerede kestiği). Aşağıdaki grafik, bu iki sayının 10 eğitim adımı boyunca değişimini gösteriyor:



Şekil 1.4: Eylem halinde öğrenme: Model, en iyi uyumu bulmak için eğitim adımları boyunca ağırlığını ve yanlışlığını ayarlıyor.

Ağırlık 4,5'ten 3'e doğru düşerken ve yanlışlık 0,4'ten 2'ye doğru yükselirken, yeşil tahmin doğrusu hedefe uyana kadar döner ve kayar:



Şekil 1.5: Eğitimden sonra: Model örüntüyü buldu ve veriye uydu.

İşte öğrenme bu. Felsefi bir şey değil. Gizemli bir şey değil. Sadece sayıları, daha iyi çıktılar üretene kadar hatalara göre ayarlamak.

Modern büyük dil modelleri yüz milyarlarca, hatta bazen bir trilyonun üzerinde parametreye sahip.

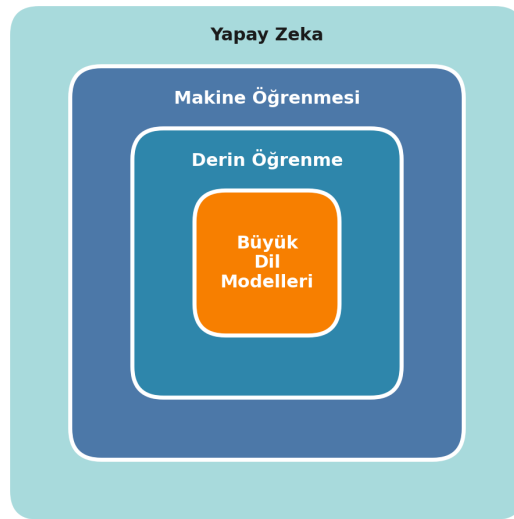
Devasa miktarda metin üzerinde trilyonlarca tahmin yaparak ayarlandılar. Sonuç: hemen hemen her bağlamda bir sonraki kelimeyi kayda değer doğrulukla tahmin edebilen bir sistem.

1.3 Yapay Zeka Aile Ağacı

“Yapay zeka” birçok farklı teknolojiyi kapsayan geniş bir terim. Hadi bunları bir sıralayalım.

1.3.1 Hiyerarşi

Bunu iç içe geçmiş daireler gibi düşün; her biri bir sonrakinin içinde:



Şekil 1.6: Yapay Zeka, Makine Öğrenmesi, Derin Öğrenme ve Büyük Dil Modellerinin iç içe hiyerarşisi.

Her katmanı tanımlayalım:

Yapay Zeka (Artificial Intelligence, AI): En geniş kategori. Bir insan yapsaydı “akıllı” olarak değerlendireceğimiz görevleri yerine getiren herhangi bir bilgisayar sistemi. Spam filtreni, satranç programlarını, öneri sistemlerini ve evet, ChatGPT’yi içeriyor.

Makine Öğrenmesi (Machine Learning, ML): Sistemlerin elle kodlanmış kurallara uymak yerine veriden öğrendiği yapay zekanın bir alt kümesi. Bir programcının “eğer X ise Y” yazması yerine, sistem kendi örüntülerini örneklerden keşfeder.

Derin Öğrenme (Deep Learning, DL): Beyindeki nöronların nasıl bağlandığından gevşekçe esinlenen “sinir ağları” sistemlerini kullanan makine öğrenmesinin bir alt kümesi. “Derin” kelimesi, bu ağların birçok katmana sahip olduğu anlamına gelir ve giderek daha soyut örüntüler öğrenmelerine olanak tanır. Örneğin görüntü tanımada, erken katmanlar çizgileri ve eğrileri algılar, orta katmanlar göz gibi şekilleri tanır, derin katmanlar ise “bu bir yüz” sonucuna varır. Derin öğrenme; görüntü tanıma, konuşma işleme ve birçok başka uygulamaya güç verir. Büyük dil modelleri, dile odaklanan bir daldır.

Büyük Dil Modelleri (Large Language Models, LLM): Özellikle dil için tasarlanmış derin öğrenmenin bir alt kümesi. Bu modeller bir dizideki bir sonraki kelimeyi tahmin etmek için eğitilir. Milyarlarca parametreye sahip oldukları ve devasa miktarda metin üzerinde eğitildikleri için “büyük” diyoruz.

1.3.2 Kısa Bir Zaman Çizelgesi

Yapay zekanın dramatik iniş çıkışlarla dolu uzun bir geçmişi var:

Yıl	Olay
1950	Alan Turing makine zekası için “Turing Testi”ni öneriyor
1956	“Yapay Zeka” terimi Dartmouth Konferansı’nda ortaya atılıyor
1960’lar-70’ler	Erken iyimserlik, sonra ilerleme durduğunda ilk “Yapay Zeka Kışı”
1997	IBM’in Deep Blue’su dünya satranç şampiyonu Garry Kasparov’u yeniyor
2012	AlexNet, ImageNet yarışmasını kazanıyor ve derin öğrenme devrimini ateşliyor
2017	Google “Attention Is All You Need” makalesini yayınlıyor, Transformer mimarisi
2018	GPT-1 yayınlanıyor (117 milyon parametre)
2020	GPT-3 yayınlanıyor (175 milyar parametre), yetenekleri araştırmacıları şaşırtıyor
2022	ChatGPT başlatılıyor, yapay zeka ana akım bilince giriyor
2023+	Hızlı ilerleme devam ediyor, birkaç ayda bir yeni modeller yayınlanıyor

2017 Transformer makalesi bu kitap için özellikle önemli. İnşa etmeyi öğreneceğin mimari bu. Her modern büyük dil modelinin temelidir.

1.4 Büyük Dil Modellerini Özel Yapan Nedir?

Dil bilgisayarlar için her zaman zor olmuştur. Neden? Çünkü dil belirsiz, bağlamsal ve sürekli gelişiyor.

“Ali’yi bankada gördüm” cümlesini düşün: - Para çekerken mi gördün? - Nehir kenarındaki sırada mı oturuyordu?

İnsanlar bu belirsizliği bağlamı kullanarak anında çözer. Bilgisayarlar ise geleneksel olarak zorlandı çünkü bağlamı temsil etmenin bir yolu yoktu.

Büyük dil modelleri bu sorunu çözdü. Transformer mimarisini kullanarak (Bölüm 3 ve Bölüm 4'te öğreneceğin gibi), her kelimeyi işlerken tüm paragraf bağlamını dikkate alabilirler. Bu sayede belirsizliği ele alabilir, referansları takip edebilir ve uzun pasajlar boyunca tutarlılığı koruyabilirler.

1.4.1 Büyük Dil Modelleri Neler Yapabilir

Modern büyük dil modelleri oldukça çok yönlü:

- **Metin yazma ve düzenleme:** Denemeler, e-postalar, raporlar, yaratıcı kurgu
- **Kod üretme:** Düzinelerce dilde çalışan programlar
- **Dilleri çevirme:** Birçok dil çiftinde iyi sonuçlar veriyor (kalite değişkenlik gösterse de)
- **Belgeleri özetleme:** Uzun metinleri anahtar noktalara yoğunlaştırma
- **Soruları yanıtlama:** Geniş eğitim bilgisinden yararlanma
- **Problemlerde muhakeme:** Çok adımlı mantık (her zaman güvenilir olmasa da)
- **Kişilikleri benimseme:** Farklı davranmak için talimatları takip etme
- **Analiz ve açıklama:** Karmaşık konuları parçalara ayırma

Dikkat çekici olan, kimsenin bu yetenekleri açıkça programlamamış olması. Hepsi tek bir basit eğitim hedefinden ortaya çıktı: bir sonraki kelimeyi tahmin et.

Araştırmacılar buna “beliren yetenek” diyor: basit kurallar karmaşık davranış ürettiğinde ortaya çıkan şey. Bir sonraki kelimeyi tahmin etmenin neden muhakeme gibi görünen bir şeye yol açtığını kimse tam olarak anlamıyor. Bunun gerçek muhakeme mi yoksa çok gelişmiş örüntü eşleştirme mi olduğu konusunda hala aktif tartışmalar var. Alandaki en büyüleyici açık sorulardan biri.

1.4.2 Büyük Dil Modelleri (Kendi Başlarına) Neler Yapamaz

Anlaşılması gereken önemli bir şey var: büyük dil modelinin kendisi sadece bir bileşen. ChatGPT veya Claude kullandığında, bir büyük dil modeli etrafında inşa edilmiş ve üzerine ek araçlar eklenmiş bir *ürün* kullanıyorsun.

Temel büyük dil modeli şunları yapamaz: - **Gerçek zamanlı bilgiye erişmek:** Sadece eğitim verisinde olanları biliyor - **Önceki konuşmaları hatırlamak:** Her oturum yeni başlıyor - **Dünyada eylemler gerçekleştirmek:** Sadece metin üretiyor - **Güvenilir matematik yapmak:** Hesaplama yerine örüntü eşleştiriyor - **Ne zaman yanıldığını bilmek:** Öz farkındalığı yok

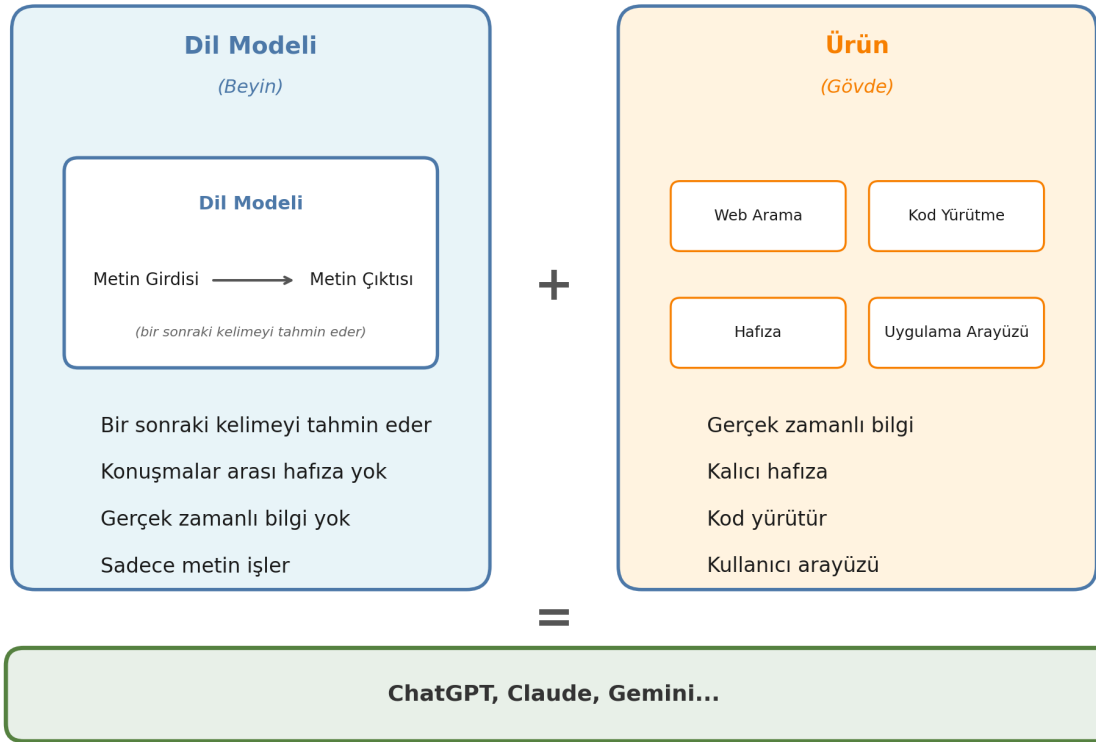
Ancak modern yapay zeka ürünleri bu sınırlamaları aşmak için araçlar ekliyor: - **Web araması:** ChatGPT güncel bilgi için internette arama yapabiliyor - **Kod yürütme:** Gerçek hesaplamalar yapmak için Python çalıştırabiliyor - **Bellek sistemleri:** Bazı ürünler konuşmalar arasında ayrıntıları hatırlıyor - **Belge alma:** Yüklenen dosyalarda veya veritabanlarında arama yapabiliyor

Bu ayrım önemli. ChatGPT sana bugünün havasını verdiğinde, büyük dil modeli havayı “bilmiyor.” Aramak için bir araç kullanıyor, sonra bulduğunu bildiriyor. Büyük dil modeli beyin gibi; araçlar ise ona el, göz ve telefon vermek gibi.

Bölüm 4'te, bu yetenekleri RAG (Retrieval-Augmented Generation, Geri Getirme ile Zenginleştirilmiş Üretim) ve araç entegrasyonu gibi teknikler kullanarak kendin eklemeyi öğreneceksin.

Temel Büyük Dil Modeli vs. Ürün

Büyük Dil Modelinin Kendisi	Ürünler Tarafından Eklenen Araçlar
Eğitime dayalı metin tahmin eder	Güncel bilgi için web araması
Oturumlar arası bellek yok	Süreklilik için bellek sistemleri
Güvenilir hesaplama yapamaz	Matematik için kod yorumlayıcı
Gerçekleri doğrulayamaz	Doğruluk için belge alma
Eylem gerçekleştirmez	Eylemler için API entegrasyonları



Şekil 1.7: Büyük dil modeli metin tahmin eder, ürün araçlar, bellek ve arayüz ekler. Birlikte ChatGPT, Claude veya Gemini olurlar.

1.4.3 Büyük Dil Modelleri Neden “Halüsinasyon Görür”?

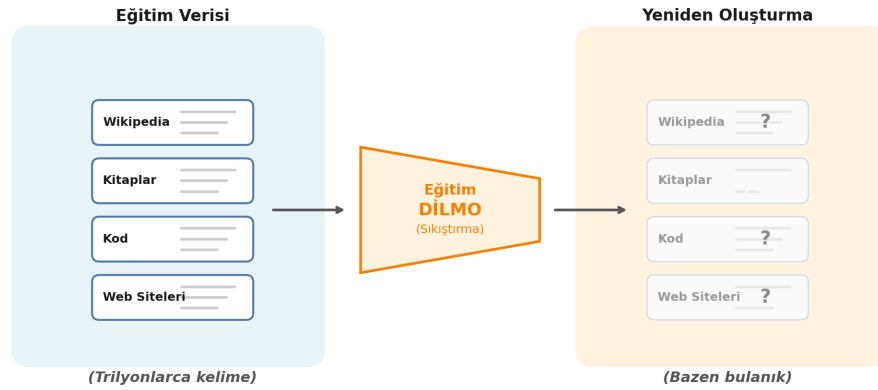
Muhtemelen büyük dil modellerinin bazen yanıltıcı şeyler uydurduğunu, yanlış bilgiyi tam bir güvenle söylediğini duymuşsundur. Buna “halüsinasyon” denir.

Neden oluyor? Çünkü büyük dil modellerinin bir gerçekler veritabanı yok. Sadece metindeki örüntülere sahipler.

“Kayıplı Sıkıştırma” Benzetmesi

OpenAI'nin kurucu üyesi ve Tesla'da eski Yapay Zeka Direktörü Andrej Karpathy, bir büyük dil modelini internetin “kayıplı sıkıştırması” olarak tanımlıyor.

Tüm interneti alıp küçük bir dosyaya sıkıştırmaya çalıştığını hayal et. Her kelimeyi olduğu gibi tutamazsın, yeterli alan yok. Bunun yerine *özü*, örüntüleri, genel fikirleri tutuyorsun. - **Sıkıştırma**: Model genel kuralları öğreniyor (dilbilgisi, muhakeme, dünya hakkında gerçekler). - **Kayıplı**: Tam ayrıntıları kaybediyor.



Şekil 1.8: Kayıplı Sıkıştırma: İnterneti bir modele sıkıştırmak “bulanık” bir yeniden yapılanma ile sonuçlanıyor.

Ona bir soru sorduğunda, o sıkıştırılmış bilgiyi anında “açıyor.” Genellikle bilgiyi doğru şekilde yeniden inşa eder. Ama bazen, sıkıştırma “kayıplı” olduğu için, yeniden yapılanma biraz sapıyor. Boşlukları kaydettiği örüntülere dayalı olarak makul *görünen* şeylerle dolduruyor. İşte halüsinasyon bu: bulanık bir bellekten kaynaklanan bir yeniden yapılanma hatası.

“Tüm modeller yanlıştır, ama bazıları yararlıdır.” — **George Box**, İstatistikçi

Box istatistiksel modellerden bahsediyordu ama bu söz büyük dil modellerine de mükemmel şekilde uyuyor. Bunlar dil modelleri, hem de gelişmiş olanları, ama yine de model. Gerçeği olduğu gibi yakalamıyorlar; genellikle gerçekle ilişkili örüntüleri yakalıyorlar. İşte bu yüzden yararlılar. Ama önemli bir şeyi doğrulaman gerektiğinin nedeni de bu.

1.5 Uygulamalı Alıştırmalar

Yapay zeka hakkında okumak seni ancak belirli bir yere kadar götürür. Hadi uygulamalı olalım.

ChatGPT’ye (veya Claude ya da başka bir büyük dil modeline) erişimin varsa, bu komutları dene ve ne olduğunu gözlemler. Yanlış cevap yok, keşfediyorsun.

1.5.1 Deneyebileceğin Komutlar

1. Basitleştirme

Fotosentezi 5 yaşındaki bir çocuğa açıklar gibi anlat.

Kelime dağarcığını nasıl uyarladığına ve benzetmeleri nasıl kullandığına dikkat et.

2. Yaratıcı yazı

Kod hata ayıklama hakkında bir haiku yaz.

5-7-5 hece yapısını takip ettiğine dikkat et.

3. Rol yapma

Sen bir korsan kaptanısın. Bileşik faizin nasıl çalıştığını açıkla.

Karmaşık bir konuyu açıklarken kişiliği nasıl sürdürdüğünü gör.

4. Muhakeme

5 makinenin 5 widget yapmak için 5 dakikaya ihtiyacı varsa, 100 makinenin 100 widget yapmak için ne

Bu klasik bir hile sorusu. Büyük dil modeli doğru bulabiliyor mu?

5. Araç kullanım testi

Dün hangi önemli haber olayları gerçekleşti?

Büyük dil modelinin web araması varsa, güncel bilgi aramasını izle. Yoksa reddetmeli veya bilmediğini kabul etmeli. Bu, temel model ile eklenen araçlar arasındaki farkı gösteriyor.

6. Öz farkındalık

Yapamadığın şeyler neler? Sınırlamaların hakkında dürüst ol.

Kendi kısıtlamalarını nasıl tanımladığına bak.

1.5.2 Alıştırma

Alıştırma 1: Bir Hata Bul

Büyük dil modeline iyi bildiğin bir konu hakkında sor. Yanlış yaptığı bir şey bulabilir misin? İnce bir hata veya tam bir uydurma olabilir. Bu teknolojiyi eleştirmek için değil. Yapay zekaya ne zaman güveneceğin ve ne zaman doğrulayacağın konusunda sezgini geliştirmek için.

Alıştırma 2: Varyasyon

Aynı soruyu üç ayrı konuşmada üç kez sor. Yanıtlar ne kadar farklı? Bu, büyük dil modellerinin olasılıksal olduğunu anlamana yardımcı olur: sabit cevaplar vermezler.

Alıştırma 3: Sınırları Zorla

Büyük dil modelinin zorlandığı görevleri bulmaya çalış: - Karmaşık çok adımlı matematik (mümkünse kod yürütme olmadan dene) - Çok belirsiz veya özel konular - Kesin gerçekseldir doğruluk gerektiren görevler - Örüntü eşleştirmeyi tuzağa düşürmek için tasarlanmış mantık bulmacaları

Bulduklarını belgele. Nerede başarılı? Nerede başarısız? Üründe web araması veya kod yürütme gibi araçlar varsa, farkı görmek için aynı görevleri bu araçlarla ve araçsız dene.

“Bana söyle, unuturum. Bana öğret, hatırlarım. Beni dahil et, öğrenirim.” — **Benjamin Franklin**

1.6 Neden Bir Tane İnşa Etmeyi Öğrenmeli?

Merak edebilirsin: ChatGPT zaten varsa, neden bir büyük dil modeli inşa etmeyi öğrenelim?

Haklı bir soru. Ama şunu düşün: tamirciler arabaları sürücülerden daha iyi anlar. Aerodinamiği anlayan pilotlar acil durumlarda daha iyi kararlar verir. Gıda kimyasını anlayan şefler sadece tarifleri takip edenlerden daha iyi yemekler yaratır.

İç yapıyı anlamak, bir teknolojiyle nasıl etkileşime girdiğini değiştirir. İşte yapay zeka için neden önemli:

1.6.1 Gizemden Arındırma

Şu anda yapay zeka sana muhtemelen sihir gibi geliyor. Bu kitabın sonunda öyle gelmeyecek. Her bileşeni, her matematiksel işlemi, her eğitim adımını anlayacaksın.

Bu önemli çünkü sihir korku ve abartıyı davet eder. Anlayış ise iyi yargıyı.

Bir toplantıda olduğunu ve birinin bir proje için yapay zeka kullanmayı önerdiğini hayal et. Diğer herkes başını sallıyor, uygulanabilir mi yoksa abartı mı emin değil. Ama sen biliyorsun. Büyük dil modellerinin gerçekte neler yapıp yapamayacağını, nerede başarılı olup nerede başarısız olduklarını anlıyorsun. Teknolojinin gerçekte nasıl çalıştığına dayalı gerçekçi bir değerlendirmeye konuşuyorsun.

İşte yapay zekayı kullanmak ile anlamak arasındaki fark bu.

1.6.2 Kariyer Avantajı

İş piyasası hızla değişiyor. Yapay zeka becerileri sektörler arası talep görüyor: - Makine öğrenmesi sistemlerini anlayan yazılım mühendisleri - Yapay zeka ekipleriyle çalışabilen ürün yöneticileri - Yapay zeka yeteneklerini ve sınırlamalarını anlayan tasarımcılar - Teknolojiyi ileri götürebilen araştırmacılar - Yapay zeka destekli ürünler inşa eden girişimciler

Fayda sağlamak için makine öğrenmesi araştırmacısı olmana gerek yok. Büyük dil modellerinin nasıl çalıştığını anlamak seni neredeyse her teknik rolde daha etkili kılıyor.

Ama iş unvanlarının ötesinde de bir şey var: dönüştürücü bir teknolojiyi erken anlayan insanlar zamanla birikimli bir avantaj kazanır. 1995'te interneti anlayanlar endüstrileri yeniden şekillendiren şirketler kurdular. Yapay zekayı şimdi anlayanlar benzer bir konumda.

1.6.3 Özelleştirme

Raftan hazır büyük dil modelleri genel amaçlı. Ama belirli bir şeye ihtiyacın olabilir: - Şirketinin belgelerinde ince ayar yapılmış bir model - Telefonda çalışan daha küçük bir model - Belirli bir alan

için özel bir asistan - Sınırlı eğitim verisine sahip bir dil veya alanda çalışan bir sistem

Temelleri anlamak, başka birinin ihtiyacın olanı inşa etmesini beklemek yerine, modelleri ihtiyaçlarına göre özelleştirmene, ince ayar yapmana ve uyarlamana olanak tanır.

1.6.4 İnşa Ederek Anlama

Şeylerin nasıl çalıştığını anlamak konusunda derin bir tatmin var. ChatGPT'ye bir soru sorduğunda ve akıllıca yanıt verdiğinde, perde arkasında *tam olarak* ne olduğunu bileceksin. Gizem mekanîğe dönüşür.

Kendi modelini eğittiğinde ve ilk kez tutarlı metin ürettiğini izlediğinde (yarattığına yardım ettiğin matematiksel örüntülerden gelen metin) gerçekten heyecan verici bir an. Bu, bir sihirbazlık numarası izlemekle nasıl yapıldığını bilmek arasındaki fark gibi.

Efsanevi fizikçi Richard Feynman, kara tahtasında bir not tutuyordu: “Yaratamazsam, anlamam.”

Bu felsefe bugün **Karpathy** tarafından savunuluyor; çalışmaları binlerce mühendise (bu kitabın yazarı olan ben dahil) ilham verdi. Onun “Sıfırdan Kahramana” serisi, sinir ağlarını temelden anlamak için altın standarttır. Kendi büyük dil modelini inşa ederek bu geleneği takip ediyorsun: gizemi ustalığa dönüştürüyorsun.

Daha Derine Girin: Bu kitapla birlikte ham kod uygulamasını görmek istersen, Andrej Karpathy'nin YouTube serisi “Neural Networks: Zero to Hero”yu şiddetle tavsiye ederim. Bu kitap, o derslerin mükemmel arkadaşı olacak şekilde tasarlandı; kavramları genişletiyor, daha fazla bağlam ekliyor ve seni üretime hazır bir sistem inşa etmenin pratiklerinde yönlendiriyor.

1.7 Kontrol Noktası Alıştırması

Süre: 15-20 dakika

Talimatlar: Geçen hafta içinde etkileşime girdiğin 5 yapay zeka sistemini belirle. Her biri için:

1. Sistem/ürünü adlandır
2. Ne yapıyor?
3. Yapay zeka aile ağacında nereye oturuyor (AI - ML - DL - LLM)?
4. Hangi örüntüleri tanıyor?

Başlamana yardımcı olacak örnekler:

Sistem	Ne yapıyor	Kategori	Tanınan örüntüler
Gmail spam filtresi	E-postaları sıralar	ML	Spam vs. meşru e-posta örüntüleri
Netflix önerileri	Diziler öneriyor	ML	İzleme tercihleri ve benzer kullanıcılar
Siri/Alexa	Sesli asistan	DL (Klasik) / LLM (Modern)	Konuşma örüntüleri, metinden niyet

Sistem	Ne yapıyor	Kategori	Tanınan örüntüler
Telefonda otomatik tamamlama	Sonraki kelimeyi tahmin eder	ML/DL	Bağlamdaki kelime dizileri

Şimdi kendin 5 tane bul!

1.8 Temel Çıkarımlar

Öğrendiklerin:

1. Yapay zeka büyük ölçekte örüntü tanımadır: istatistik + veri + hesaplama gücü
2. Makine öğrenmesi sistemleri geri bildirimle dayalı parametreleri ayarlayarak öğrenir
3. Büyük dil modelleri hiyerarşiye oturur: AI → ML → Derin Öğrenme → LLM
4. Büyük dil modelleri yazabilir, kodlayabilir, çevirebilir ve muhakeme edebilir, ancak kendi başlarına gerçek zamanlı bilgiye erişemez veya güvenilir hesaplama yapamaz
5. Halüsinasyon, büyük dil modellerinin eğitim verisinin “kayıplı sıkıştırması” olduğu için gerçekleşir
6. Modern yapay zeka ürünleri temel büyük dil modeli etrafına araçlar ekler (web araması, kod yürütme)

Temel kavramlar:

- **Örüntü tanıma:** İnsanların manuel olarak kodlamak için çok karmaşık olan verideki örüntüleri bulma
- **Parametreler/Ağırlıklar:** Eğitim sırasında ayarlanan sayılar (GPT-3’ün 175 milyarı var)
- **Beliren yetenek:** Basit eğitim hedeflerinden ortaya çıkan karmaşık yetenekler
- **Halüsinasyon:** Devasa bilgiyi örüntülere sıkıştırmaktan kaynaklanan yeniden yapılanma hataları
- **Temel Büyük Dil Modeli vs Ürün:** Model metin tahmin eder, araçlar ona gerçek dünya yetenekleri verir

Yapay zeka hiyerarşisi:

Yapay Zeka (en geniş)

 Makine Öğrenmesi (veriden öğrenir)

 Derin Öğrenme (çok katmanlı sinir ağları)

 Büyük Dil Modelleri (sonraki kelimeyi tahmin eder)

i İnceleme Sorusu Cevapları

Bu inceleme sorularının tüm cevapları **Ek D**’de mevcuttur.

1.9 İnceleme Soruları

1. Yapay zekanın en basit tanımı nedir?
 2. Makine Öğrenmesi geleneksel programlamadan nasıl farklıdır?
 3. Büyük dil modelleri yapay zeka aile ağacında nereye oturur?
 4. Büyük dil modellerinin iyi olduğu üç şeyi ve iyi yapamadığı üç şeyi adlandırın.
 5. Büyük dil modellerinde halüsinasyon neden gerçekleşir?
 6. Bir yapay zeka araştırmacısı olmayı planlamasan bile, büyük dil modellerinin nasıl çalıştığını anlamak neden değerli olabilir?
-

1.10 Sırada Ne Var

Artık yapay zekanın ne olduğunu ve olmadığını anlıyorsun. Büyük dil modellerinin manzarada nereye oturduğunu ve onları özel yapan şeyi biliyorsun.

Ama üst düzeyde konuşuyorduk. Bir bilgisayar metinle gerçekte nasıl çalışır? Sadece sayıları anlar. Kelimelerden anlamı yakalayan bir şekilde sayılara nasıl gideriz?

İşte Bölüm 2: Bilgisayarlar Kelimeleri Nasıl “Anlar”. Metin nasıl matematiğe dönüşür, her şeyin üzerine inşa edildiği temeli keşfedeceğiz.

Bölüm 2

Bilgisayarlar Kelimeleri Nasıl “Anlar”?

“Dilimin sınırları, dünyamın sınırları anlamına gelir.” — **Ludwig Wittgenstein**, Filozof

Neler Öğreneceksin - Kelimeleri temsil etmek için basit yaklaşımların (ASCII gibi) neden işe yaramadığını - Kelimelerin matematiksel bir uzayda konumlar olarak nasıl temsil edilebildiğini - Gömme (embedding) kavramının ne olduğunu ve neden bu kadar güçlü olduğunu - “Kral - adam + kadın = kraliçe” denkleminin neden gerçekten işe yaradığını - Makinelerin kelime anlamlarını bağlamdan nasıl öğrendiğini

Temel Terimler

- *Gömme (Embedding)*: Bir kelimenin anlamını yakalayan sayılardan oluşan vektör gösterimi [Terimler sözlüğüne bak](#)
- *Vektör (Vector)*: Uzayda bir konumu temsil eden sayılar listesi
- *Boyut (Dimension)*: Gömme uzayındaki bir eksen; daha fazla boyut daha nüanslı temsillere olanak tanır [Terimler sözlüğüne bak](#)
- *Kelime Uzayı (Word Space)*: Kelime gömmelerinin bulunduğu matematiksel uzay; mesafeler anlamsal farklılıklara karşılık gelir
- *Dağılımsal Hipotez (Distributional Hypothesis)*: Benzer bağlamlarda görünen kelimelerin benzer anlamlara sahip olduğu fikri
- *Anlamsal Benzerlik (Semantic Similarity)*: İki kelimenin anlam olarak ne kadar ilişkili olduğu; gömme uzayında mesafe olarak ölçülebilir
- *Kosinüs Benzerliği (Cosine Similarity)*: İki vektörün ne kadar benzer olduğunu ölçmek için kullanılan standart yöntem; -1 ile 1 arasında değer alır [Terimler sözlüğüne bak](#)
- *Word2Vec*: 2013 yılında geliştirilen ve metinden kelime gömmelerini öğrenmek için kullanılan temel bir sistem
- *Öz-Denetimli Öğrenme (Self-Supervised Learning)*: Öğrenme sinyalinin insan etiketlerinden değil, verinin kendisinden geldiği eğitim yaklaşımı

Kontrol Noktası

Bu bölümün sonunda şunları anlayacaksın:

1. Kelimelerin anlamı yakalayan sayılara nasıl dönüştüğünü
2. Gömmelerin neden yüzlerce boyut kullandığını
3. Bağlamın makinelere anlamsal ilişkileri nasıl öğrettiğini
4. Gömmeleri pratikte nasıl görebileceğini
5. Gömmeleri bağlamsal anlayışa dönüştürmek için neden dikkate (attention) ihtiyaç duyulduğunu

Bölüm 1'de, BDM'lerin (Büyük Dil Modelleri) bir sonraki kelimeyi tahmin ettiğini belirtmiştik. Ama bu hemen akla bir soru getiriyor: bilgisayarlar yalnızca sayıları anlar. “Kedi” veya “demokrasi” veya “güzel” kelimelerini okuyamazlar, yalnızca 0'lar ve 1'lerle işlem yapabilirler.

Peki kelimelerden sayılara, *anlamı* da koruyarak nasıl geçebiliriz?

Bu bölüm tam da bu soruya yanıt veriyor. Çözüm, yapay zekanın en zarif fikirlerinden biri. Ve bunu anlamak, bundan sonraki her şey için temel oluşturuyor.

2.1 Bilgisayarın İkilemi

Yalnızca sayılarla konuşan bir arkadaşına bir şeyler öğretmeye çalıştığını hayal et. “Kedi” kavramını açıklamak istiyorsun. Bunu nasıl yaparsın?

İlk içgüdün harfleri sayılara dönüştürmek olabilir. Sonuçta bilgisayarlar metni bu şekilde saklıyor.

2.1.1 ASCII Yaklaşımı (Ve Neden Başarısız Olur)

Klavyedeki her karakterin bir sayısı var. ‘k’ harfi 107, ‘e’ 101, ‘d’ 100, ‘i’ 105. Yani “kedi” kelimesi [107, 101, 100, 105] oluyor.

Sorun çözüldü mü? Hayır, hiç de değil.

Şuna bir bak: - “kedi” = [107, 101, 100, 105] - “kent” = [107, 101, 110, 116]

Bu sayılar neredeyse aynı (sadece son iki pozisyonda küçük farklar var). Oysa bir kedi ile bir kent arasında hiçbir ortak nokta yok. Biri canlı bir hayvan, diğeri bir yerleşim yeri.

Bir de şuna bak: - “kedi” = [107, 101, 100, 105] - “yavru kedi” = [121, 97, 118, 114, 117, 32, 107, 101, 100, 105] - “kedicik” = [107, 101, 100, 105, 99, 105, 107]

Bu sayılar tamamen farklı görünüyor, oysa üç kelime de aslında aynı yaratığı ifade ediyor.

ASCII kodları *yazımı* yakalar, *anlamı* değil. Üstelik yazım keyfi bir şey: “kedi”nin k-e-d-i olarak yazılması için özel bir neden yok, başka bir şey de olabilirdi.

2.1.2 Kelime Kimliği Yaklaşımı (Ve Neden O da Başarısız Olur)

Tamam, harfleri unuttuk. Peki ya her kelimeye benzersiz bir kimlik (ID) atarsak?

- kedi = 1
- köpek = 2
- balık = 3
- kedigil = 4
- yavru kedi = 5
- otomobil = 6

Şimdi “kedi” ve “kent” farklı sayılara sahip. Ama yeni bir sorun yarattık: bu sayıların birbirleriyle hiçbir ilişkisi yok.

Bu sistemde “kedi” (1) ile “yavru kedi” (5) arasındaki uzaklık, “kedi” (1) ile “otomobil” (6) arasındaki uzaklık kadar. Sayılar, kedilerin ve yavru kedilerin benzer olduğunu, kedilerle otomobillerin benzer olmadığını yakalamıyor.

Bu kimliklerle matematik yapabiliriz: kedi + köpek = 3. Ama bu balığa eşit, ki matematiksel bir saçmalık. Bu da kimliklerin ilişkileri ne kadar kötü yakaladığını gösteriyor.

2.1.3 Gerçekte Neye İhtiyacımız Var

Aslında ihtiyacımız olan şey şu: kelimeleri sayılara öyle dönüştürmeli ki:

1. Benzer kelimeler benzer sayılar alsın
2. İlişkisiz kelimeler farklı sayılar alsın
3. Kelimeler arasındaki ilişkiler korunsun

Bunun mümkün olduğu ortaya çıktı. Ama sayıları biraz farklı düşünmemiz gerekiyor.

2.2 Konum Olarak Kelimeler

İşte kilit içgörü: her kelimeye tek bir sayı vermek yerine, her kelimeye bir *konum* verelim.

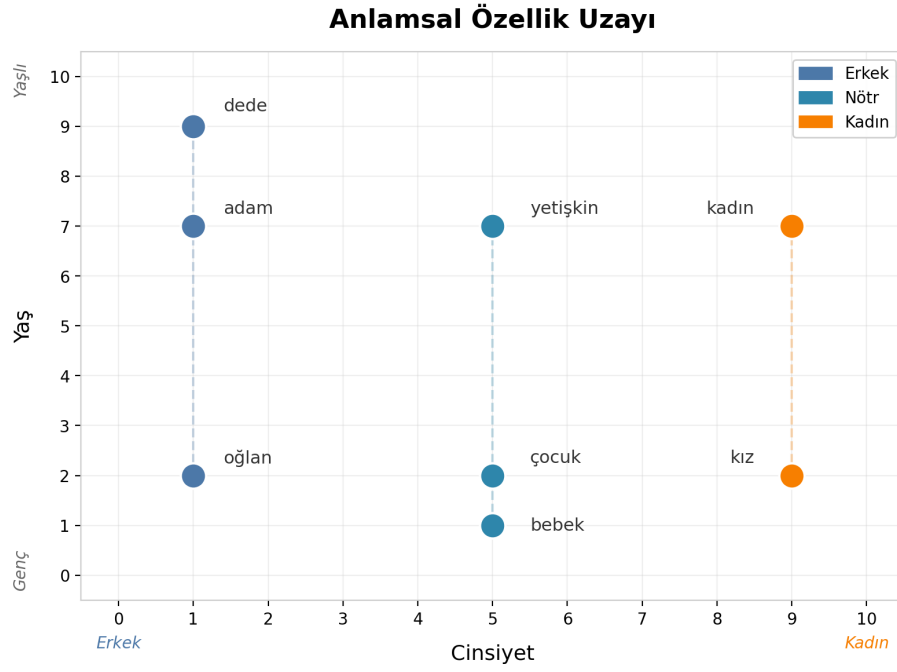
Bir şehir haritasını düşün. Haritada her konumun koordinatları var: bir X konumu ve bir Y konumu. Birbirine yakın iki restoran benzer koordinatlara sahip. Şehrin karşı taraflarında bulunan bir restoran ve bir park ise çok farklı koordinatlara sahip.

Peki ya kelimeler için de aynı şeyi yapsak?

2.2.1 Kelime Uzayı

İki boyutlu bir uzay hayal et, milimetrik kağıt gibi. Şimdi kelimeleri anlamlarına göre bu kağıdın üzerine yerleştirdiğini düşün:

Bu görselleştirmede:



Şekil 2.1: Kelimelerin 2 boyutlu anlamsal uzayda çizilmesi, konumun anlamı yansıttığı yer. Benzer kelimeler bir araya kümelenir.

- **Cinsiyet Eksen (X eksen):** Kelimeler erkekten (sol) kadına (sağ) doğru kayıyor. “Adam” ve “kadın”ın bu ekseninde birbirinden uzak olduğuna, ama benzer yüksekliklerde olduğuna dikkat et.
- **Yaş Eksen (Y eksen):** Kelimeler gençten (alt) yaşlıya (üst) doğru kayıyor. “Çocuk” altta, “adam” ortada, “büyükbaba” üstte.
- **Kümelenme:** Benzer anlamsal özelliklere sahip kelimeler benzer konumlarda yer alıyor. “Çocuk” ve “kız” aynı yükseklikte (aynı yaş) ama cinsiyet ekseninin karşı taraflarında. “Adam” ve “kadın” da aynı kalıbı paylaşıyor: aynı yaş, farklı cinsiyet.

(Not: Bu basitleştirilmiş görselleştirmede, kelimeleri Cinsiyet ve Yaş gibi yorumlanabilir boyutlara eşledik. Gerçek gömmeler ise eğitim verisinden otomatik olarak ortaya çıkan yüzlerce boyut kullanır. Bu boyutlar nadiren net insan kavramlarına karşılık gelir; matematiksel olarak yararlı ama genellikle adlandırılmayan istatistiksel kalıpları yakalarlar.)

Bir kelimenin *konumu* anlamı hakkında bilgi taşır. Benzer anlamlar benzer konumlara yol açar.

2.2.2 Koordinatlar Anlamı Yakalar

2 boyutlu bir uzayda her kelimenin iki sayısı var (X ve Y koordinatları):

- “çocuk” [1.0, 2.0] konumunda olabilir (Cinsiyet=1, Yaş=2)
- “adam” [1.0, 7.0] konumunda olabilir (Cinsiyet=1, Yaş=7)
- “kadın” [9.0, 7.0] konumunda olabilir (Cinsiyet=9, Yaş=7)

i Matematik Notu: Vektör Nedir?

Vektör, uzayda bir konumu temsil eden sayılar listesinden başka bir şey değil. [1.0, 2.0] listesi 2 boyutlu bir vektör. Matematik detayları için endişelenme; vektörleri daha derinlemesine anlamak istersen **Ek C.1**’e bakabilirsin (Matematik Tazeleme: Skalerler, Vektörler ve Matrisler).

Kilit içgörü şu: **bu uzayda iki kelime ne kadar yakınsa, anlamları o kadar benzer.**

Bunu matematiksel olarak ölçebiliriz. Bu 2 boyutlu ızgarada “çocuk”, “kadın”a (yaklaşık 9.4 mesafe) göre “adam”a (mesafe 5) daha yakın.

Gerçek yüksek boyutlu gömmelerde standart yaklaşım, vektörler arasındaki açıyı ölçen **kosinüs benzerliğini** kullanmak. Benzer yönleri gösteren iki kelime benzer demek.

i Matematik Notu: Kosinüs Benzerliği Nasıl Çalışır?

Kosinüs benzerliği, iki vektörün ne kadar benzer olduğunu, mutlak mesafelerine değil aralarındaki açığa bakarak ölçer. Bu özellik, onu kelime anlamlarını karşılaştırmak için ideal kılıyor. Gerçek hesaplamayı görmek istersen **Ek C.4**’e bakabilirsin (Matematik Tazeleme: İç Çarpımlar ve Benzerlik).

Artık anlam ilişkilerini yakalayan sayılarımız var.

2.2.3 Ancak İki Boyut Yeterli Değil

Elbette iki boyut dilin tüm nüanslarını yakalayamaz. Şu kelimeleri düşün:

- “kral” ve “kraliçe” (ikisi de kraliyet)
- “kral” ve “adam” (ikisi de erkek)
- “kral” ve “başkan” (ikisi de lider)

Bir kral, bir yönden kraliçeye (kraliyet), başka bir yönden adama (cinsiyet), yine başka bir yönden başkana (liderlik) benzer. Tüm bu farklı benzerlik türlerini yakalamak için daha fazla **boyuta** ihtiyacımız var.

Peki ya yüzlerce boyutumuz olsaydı? Ya da binlerce?

2.2.4 Daha Yüksek Boyutları Görselleştirme

768 boyutu hayal etmek zor, ama 3 boyutu görselleştirmeyi deneyebiliriz. Üçüncü bir eksen (Kraliyet) ekleyerek kelimeler arasındaki başka bir ilişki türünü yakalayabiliriz.

(Not: Bu 3 boyutlu diyagramda rengi yukarıdaki 2 boyutlu diyagramdan farklı kullanıyoruz. Burada renk cinsiyeti değil, kraliyet seviyesini kodluyor.)

Bu 3 boyutlu görselleştirmede, renk gradyanının üçüncü boyutu nasıl kodladığına dikkat et: mavi düşük kraliyeti (çocuk, kız, adam, kadın gibi sıradan insanlar), turuncu ise yüksek kraliyeti (kral, kraliçe, prens, prenses) gösteriyor. Uzaysal kümelenme ilişkileri görünür kılıyor: prensin krala çocuktan daha yakın olduğunu, prensesin de kraliçeye kızdan daha yakın olduğunu görebilirsin. Ve bu sadece üç boyutla. 768 boyutu görselleştiremezsin, ama matematik aynı şekilde çalışıyor: benzer anlamlara sahip kelimeler bu yüksek boyutlu uzayda birbirine komşu oluyor.

2.3 Gömmelerin Büyüsü

Modern yapay zekanın tam olarak yaptığı şey bu. İki boyut yerine, kelimeler yüzlerce veya binlerce boyutlu uzaylarda temsil ediliyor.

Bu temsil (bir kelimenin yüksek boyutlu uzaydaki konumunu kodlayan sayılar listesi) **gömme** olarak adlandırılıyor.

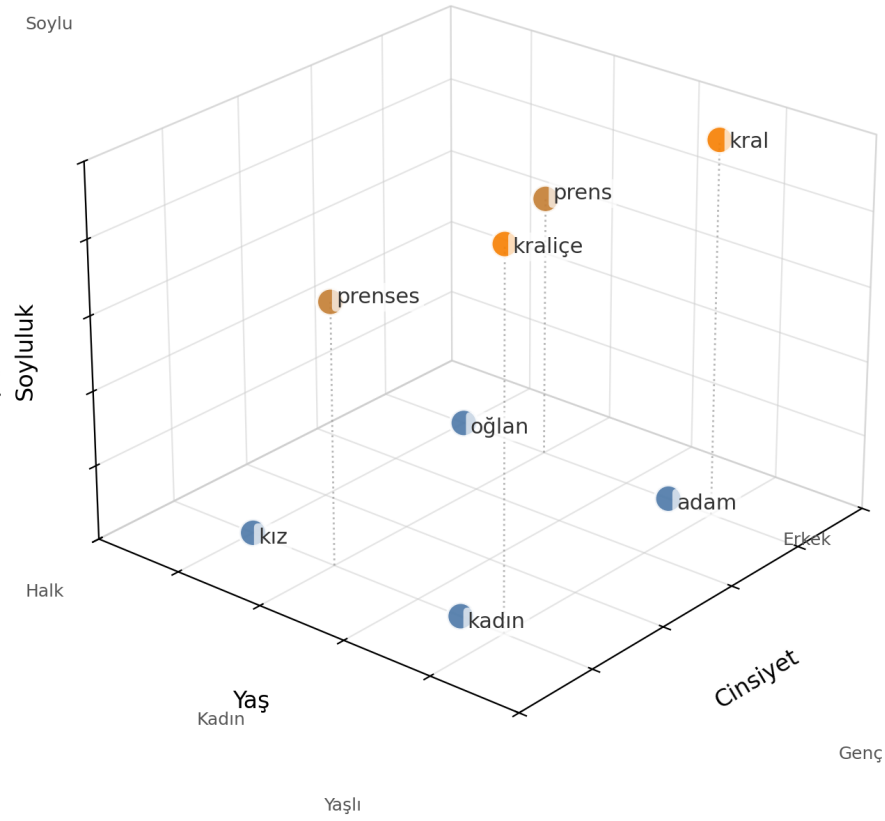
2.3.1 Bir Gömme Nasıl Görünür

İşte gerçek bir gömmenin nasıl görünebileceği (basitleştirilmiş):

"kral" = [0.82, 0.31, 0.91, -0.24, 0.55, 0.12, ...] (toplam 768 sayı)
 "kraliçe" = [0.79, 0.33, 0.88, -0.19, 0.52, 0.15, ...] (benzer!)
 "muz" = [-0.51, 0.89, 0.11, 0.63, -0.22, 0.77, ...] (çok farklı!)

“Kral” ve “kraliçe”nin her pozisyonda benzer sayılara sahip olduğuna, “muz”un ise tamamen farklı sayılara sahip olduğuna dikkat et. Bunun nedeni kralların ve kraliçelerin anlamsal olarak ilişkili olması, muzun ise ikisiyle de ilişkisiz olması.

Anlamsal Özellik Uzayı



Şekil 2.2: Üçüncü bir boyut (Kraliyet) eklemek, daha fazla anlamsal ilişkiyi yakalamaya olanak tanır. Kral ve kraliçe yüksek kraliyet paylaşıırken, çocuk ve kız sıradan insanlar olarak kalır.

Biliyor muydun? Çok Dilli Sihir

Çok dilli BDM’ler dikkat çekici bir şey keşfediyor: “dog” (İngilizce) ve “hund” (Almanca) gibi kelimeler, kimse modele bunların ilişkili olduğunu söylemeden gömme uzayında komşu oluyor. Nasıl mı? Eğitim sırasında bu kelimeler benzer bağlamlarda görünüyor (kedi kovalar, dört ayağı vardır, havlar), bu yüzden model bunların aynı kavrama atıfta bulunduğunu öğreniyor. Yani bir BDM İngilizce’de köpekler hakkında bir gerçek öğrendiğinde, bunu Almanca’da da otomatik olarak “biliyor”. Gömme uzayı dile göre değil, anlama göre düzenleniyor.

2.3.2 Daha Fazla Boyut Neden Yardımcı Olur

Bir insanı tanımlamak gibi düşün:

- **1 boyut** (boy): Uzun insanları kılalardan ayırt edebilirsin.
- **2 boyut** (boy + kilo): Artık uzun-inceden uzun-kiloluyu ayırt edebilirsin.
- **5 boyut** (yaş, saç rengi, göz rengi ekle): Çok daha spesifik olur.
- **100 boyut**: Görünümdeki ince farkları tanımlayabilirsin.

Benzer şekilde:

- Kelimeler için **2 boyut**: Kaba gruplamalar (hayvanlar vs. nesneler)
- **100 boyut**: Memelileri sürüngenlerden, evcil hayvanları yabani hayvanlardan ayırabilir
- **768 boyut**: “Gezinti”nin keyifli bir yürüyüş, “yürüyüş”ün ise kararlı bir hareket olduğu gibi ince nüansları yakalayabilir

Modern BDM’ler 768 (daha küçük modeller) ile 12.288 veya daha fazlası (büyük modeller) arasında değişen gömme boyutları kullanıyor. 768 sayısı yaygın bir standart çünkü etkili BERT-base ve GPT-2 küçük modellerinde kullanılan boyuttu. Her boyut anlamın bir yönünü yakalıyor, ama (önemli olarak) hiçbir tekil boyutun “boyut” veya “canlılık” gibi net bir insan yorumu yok. Boyutlar eğitimden ortaya çıkıyor ve anlamı dağınık, karmaşık şekillerde kodluyor.

2.3.3 Neden “Gömme”?

Terim matematiğe dayanıyor. Matematikte “gömme”, önemli yapıyı korurken nesneleri bir uzaydan diğerine eşlemek demek. Biz de ayrı sembolleri (“kedi” ve “kent”in doğal bir ilişkisi olmadığı kelimeleri) alıp sürekli bir uzaya (konumların anlamı kodladığı yere) gömüyoruz.

Kelimeler bu matematiksel uzaya gömüldükten sonra mesafeleri ölçebilir, komşuları bulabilir ve aritmetik yapabiliriz. Dil işlemeyi matematiksel yapan, dolayısıyla bilgisayarların yapabileceği bir şey haline getiren tam da bu.

2.4 Ünlü Denklemler

2013 yılında Google'daki araştırmacılar beklenmedik bir keşif yaptı. Kelime gömmeleri oluşturmak için bir sistem eğittiklerinde dikkat çekici bir şey ortaya çıktı.

Kelimelerle aritmetik yapabiliyorlardı.

2.4.1 Kral - Adam + Kadın = ?

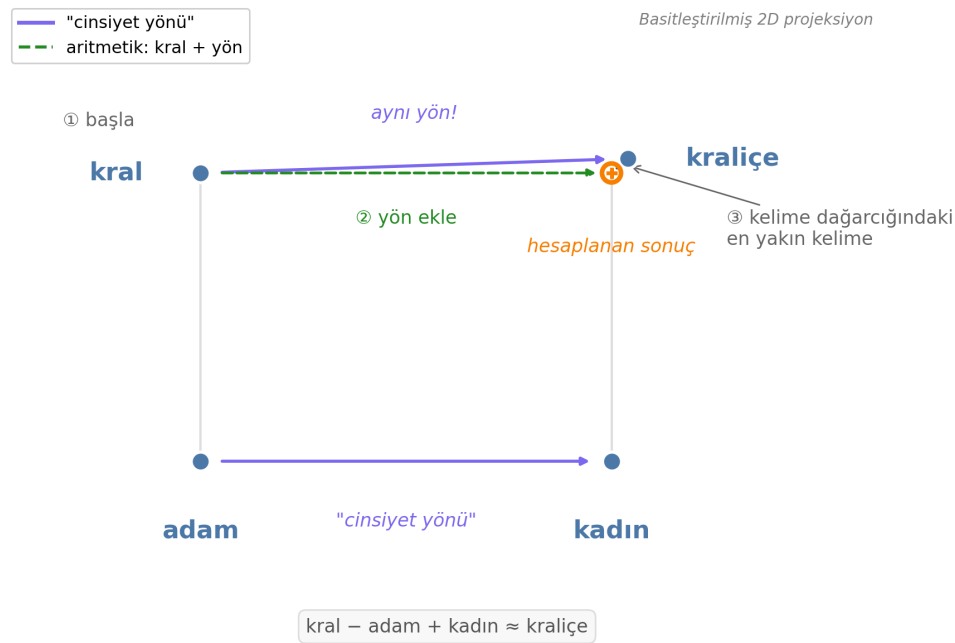
“Kral” için gömmeyi al. “Adam” için gömmeyi çıkar. “Kadın” için gömmeyi ekle. Ne elde edersin?

Sonuç uzayda bir nokta ve bu noktaya en yakın kelime dağarcığı kelimesi... “kraliçe”.

```
en_yakin_kelime( gömme("kral") - gömme("adam") + gömme("kadın") ) = "kraliçe"
```

Hesaplanan vektör tam olarak kraliçenin gömmesine eşit değil, ama kraliçe kelime dağarcığında o konuma en yakın kelime.

Şekil 2.3 bu ilişkiyi geometrik olarak gösteriyor. Mor okların (adam→kadın ve kral→kraliçe'nin “cinsiyet yönü”) nasıl paralel olduğuna dikkat et. “Adam”ı “kadın”a değiştiren aynı dönüşüm, “kral”ı da “kraliçe”ye dönüştürüyor.



Şekil 2.3: Gömme uzayında vektör aritmetiği: “kral”dan (mavi nokta) başlayarak, adam→kadın’ı dönüştüren aynı yönü uygulamak (mor ok), hesaplanmış bir sonuç (turuncu daire) üretir. O noktaya en yakın kelime dağarcığı kelimesi “kraliçe”dir (yeşil nokta).

Bunun ne anlama geldiğini düşün. Sistem, kimse söylemeden şunu öğrenmiş: - “Kral” ile “adam”

arasındaki ilişki (erkek kraliyet ile erkek) - “Kraliçe” ile “kadın” arasındaki ilişkiyle aynı (kadın kraliyet ile kadın)

Bunu kimse programlamadı. Gömmeler *dilin temel yapısını yakalamış*.

2.4.2 Daha Fazla Örnek

Bu birçok ilişki için çalışıyor (ama mükemmel değil; bu “benzetme görevleri”ndeki doğruluk genellikle %40-70 civarında, %100 değil):

Başkentler: - Paris - Fransa + İtalya $\approx$ Roma - Tokyo - Japonya + Almanya $\approx$ Berlin

Fiil zamanları: - yürüyor - yürümek + yüzmek $\approx$ yüzüyor - koştu - koşmak + uçmak $\approx$ uçtu

Karşılaştırmalar: - daha büyük - büyük + küçük $\approx$ daha küçük - en hızlı - hızlı + yavaş $\approx$ en yavaş

İlişkiler: - erkek kardeş - adam + kadın $\approx$ kız kardeş - amca - adam + kadın $\approx$ teyze

2.4.3 Bu Ne Ortaya Koyuyor

Bu kelime aritmetiği derin bir şeyi ortaya koyuyor: gömmeler sadece benzer kelimeleri bir araya getirmiyor. Kelimeleri, *ilişkilerin* tutarlı olacağı şekilde düzenliyorlar.

Gömme uzayında adamdaki kadına doğru “yön”, kraldan kraliçeye, erkek kardeştan kız kardeşe, amcadan teyzeye doğru yönle kabaca aynı. Gömmeler cinsiyet dönüşümünün soyut bir kavramını öğrenmiş.

Benzer şekilde, seni Fransa’dan Paris’e, Japonya’dan Tokyo’ya, Almanya’dan Berlin’e götüren bir “başkenti” yönü var.

2.4.4 Önemli Uyarılar

Bu her zaman mükemmel çalışmıyor. Bazen en yakın kelime tam olarak doğru değil. Bazen eğitim verisindeki önyargılar sorunlu kalıplar yaratıyor. Aritmetik yaklaşık, kesin değil.

Ama bunun herhangi bir şekilde çalışması (kelimelerle anlamlı aritmetik yapabilmen), gömmelerin dil yapısı hakkında gerçek bir şeyi yakaladığını gösteriyor. Bunlar rastgele sayılar değil; anlam haritası.

i Bağlantı: Kayıplı Sıkıştırma

Bölüm 1’deki “kayıplı sıkıştırma” benzetmesini hatırlıyor musun? Gömmeler o sıkıştırmanın *ilk* katmanı, insan dilinin sonsuz karmaşıklığını alıp sabit boyutlu vektörlere sıkıştırıyor. Tüm model daha sonra bu sıkıştırmayı dikkat ve ileri beslemeli katmanlar aracılığıyla sürdürüyor. Tıpkı sıkıştırılmış bir görüntünün bazı detayları kaybetmesi gibi, model kesin gerçekleri değil kalıpları yakalıyor. Bu nedenle BDM’ler hiç tam olarak görmedikleri kavramlar hakkında akıl yürütebiliyor, ama aynı zamanda bazen “halüsinasyon” olarak adlandırılan makul görünen ancak yanlış bilgiler de üretebiliyor.

2.5 Gömmeler Nasıl Öğrenilir?

Bu sihirli sayılar nereden geliyor? Bir sistem, “kral” ve “kraliçe”nin birbirine yakın, “kral” ve “muz”un uzak olması gerektiğini nasıl öğreniyor?

Cevap güzel bir şekilde basit: **bağlam**.

2.5.1 Bir Kelimeyi Arkadaşlarından Tanırsın

Dilbilimci J.R. Firth’ün bu sözü temel bir içgörüyü yakalıyor: benzer bağlamlarda görünen kelimeler benzer anlamlara sahip olma eğiliminde.

Şu cümleleri düşün: - “_____ paspasın üzerinde oturdu.” - “_____ fareyi kovaladı.” - “_____’nın kürkünü okşadı.”

Boşluklara hangi kelimeler uyar? Kedi, köpek, yavru kedi, köpek yavrusu. Bu kelimeler benzer bağlamlarda görünüyor, demek ki ilişkili anlamlara sahipler.

Şimdi şunları düşün:

- “_____’ya para yatırdım.”
- “_____ kredimi onayladı.”
- “_____’da gişe görevlisi olarak çalışıyor.”

Cevap “banka” (finans kurumu). Kedi cümlelerinden farklı bağlamlar, farklı anlam kümesi.

2.5.2 Tahmin Etmekten Öğrenme

Gömmeler sistemi gerçekte şöyle öğreniyor. Temel fikir (Word2Vec tarafından 2013’te öncülük edildi):

1. **Büyük miktarda metin al** (internette, kitaplardan vb. milyarlarca kelime)
2. **Bir tahmin görevi oluştur**: Bir boşluğun etrafındaki kelimeler verildiğinde, eksik kelimeyi tahmin et
3. **Gömmeleri ayarla** öyle ki benzer bağlamlardaki kelimeler benzer konumlar alsın

Örneğin “_____ paspasın üzerinde oturdu” verildiğinde sistem “paspas”, “zemin”, “kanepe” vb.’nin muhtemel olduğunu tahmin etmeli. “Banka _____ onayladı” verildiğinde ise “kredi”, “ipoteke”, “başvuru” tahmin etmeli.

Milyarlarca tahmin sayesinde sistem şunları öğreniyor:

- “Kedi” ve “köpek” benzer bağlamlarda görünüyor → benzer gömmeler alıyorlar
- Çeşitli bağlamlarda görünen kelimeler (hem “nehir” hem de “para” yakınında “banka” gibi) uzlaşmacı gömmelerle sonuçlanıyor, bu birazdan ele alacağımız bir sınırlama
- “Kral” kraliyet, liderlik, erkek hükümdarlar hakkında bağlamlarda görünüyor → gömmesi tüm bunları kodluyor

2.5.3 Öz-Denetimin Güzelliği

Dikkat çekici olan şey, bunun insan etiketlemesi gerektirmemesi. Sistemin “kedi” ve “yavru kedi”nin ilişkili olduğunu kimsenin söylemesi gerekmedi. Sistem bunu yalnızca bağlamdan anladı.

Buna **öz-denetimli öğrenme** deniyor: eğitim sinyali insan açıklamalarından değil, verinin yapısının kendisinden geliyor. Milyarlarca kelime üzerinde eğitim yapabilmemizin nedeni de bu, çünkü bu kadar veriyi elle etiketlemek imkansız olurdu.

2.5.4 Statik ve Bağlamsal Gömmeler

Açıklamamız gereken önemli bir ayrım var. Word2Vec, her kelimeye bağlamdan bağımsız olarak tek bir sabit gömme veriyor. “Banka” kelimesi, “nehir kıyısı”nda mı yoksa “banka hesabı”nda mı görüldüğüne bakmaksızın aynı gömmeyi alıyor; farklı anlamları arasında bir uzlaşma konumu.

Modern BDM’ler bunu önemli ölçüde geliştiriyor. **Bağlamsal gömmeler** üretiyorlar: aynı kelime için çevreleyen kelimelere bağlı olarak farklı gömmeler. Bir transformer modelinde “banka”, “nehir kıyısı”nda bir gömme ve “banka hesabı”nda tamamen farklı bir gömme alıyor.

Nasıl mı? **Bölüm 3’te ele alacağımız** dikkat mekanizması aracılığıyla. İlk kelime gömmesi sadece bir başlangıç noktası; kullanılmadan önce bağlama göre değiştiriliyor.

Dolayısıyla Word2Vec tarzı statik gömmeler önemli bir sıçramaydı, ama modern BDM’ler onların ötesine geçti. Statik gömmeleri anlamak sezgini oluşturuyor; dikkati anlamak ise bu sınırlamaları nasıl aştığımızı gösteriyor.

2.6 Kelimelerden Cümlelere

Bireysel kelimeleri temsil etme sorununu çözdük. Ama dil sadece kelimeler değil, cümleler, paragraflar, belgeler. Dizileri nasıl ele alırız?

2.6.1 Naif Yaklaşım: Ortalama Alma

En basit fikir: bir cümledeki tüm kelime gömmelerinin ortalamasını al.

“Kedi oturdu” $\rightarrow$ (gömme(“kedi”) + gömme(“oturdu”)) / 2

Bu sana cümleyi temsil eden tek bir vektör veriyor. Ve işe yarıyor... bir bakıma. Benzer belgeler bulmak gibi bazı görevler için ortalama alma şaşırtıcı derecede etkili.

Ama ölümcül bir kusur var.

2.6.2 Sıra Önemlidir!

Şu cümleleri düşün: - “Köpek adamı ısırıldı” - “Adam köpeği ısırıldı”

Aynı kelimeler. Aynı ortalama gömme. Tamamen farklı anlamlar.

İlki olağandışı değil (köpekler bazen ısırır). İkincisi tuhaf ve haber değeri taşıyor.

Ya da şunları düşün: - “Film iyi değildi” - “Film kötü değil, iyiydi”

Ortalama alma her iki cümle için “iyi” ve “değil” kelimelerini benzer şekillerde karıştırıyor, birinin olumsuz diğerinin olumlu olduğunu kaçırıyor.

2.6.3 Sorun

Kelime gömmeleri (en azından Word2Vec gibi statik olanlar) kelimelerin izole anlamlarını yakalıyor. Ama dildeki anlam şunlara bağlı:

- **Kelime sırası:** “Köpek adamı ısırıldı” ile “Adam köpeği ısırıldı” aynı anlama gelmiyor
- **Uzun menzilli ilişkiler:** “Paspasın üzerinde oturan kedi...” “Oturana” neye atıfta bulunuyor?
- **Bağlam değişikliği:** “İyi değil” ifadesi “iyi”nin anlamını değiştiriyor
- **Belirsizliğin giderilmesi:** “Bankaya gittim” “nehir” ve “para” yakınında farklı şeyler ifade ediyor

Ortalama alma tüm bu bilgiyi atıyor. Tam bağlama bakabilen ve kelimelerin birbirleriyle nasıl ilişkili olduğunu anlayabilen bir şey ihtiyacımız var.

2.6.4 Neye İhtiyacımız Var

İhtiyacımız olan:

1. Kelimeleri sırayla işlemek, sırayı korumak
2. Kelimelerin bağlama göre birbirlerinin anlamlarını etkilemesine izin vermek
3. Metindeki uzun mesafeli ilişkileri ele almak
4. Aynı kelime için farklı bağlamlarda farklı temsiller üretmek

Dikkat mekanizması tam olarak bunu yapıyor ve **Bölüm 3**’ün konusu da bu. Dikkat, her kelimenin diğer her kelimeye “bakmasına” ve neyin ilgili olduğuna karar vermesine olanak tanıyor. Gömmeleri iyi bir fikirden modern yapay zekanın temeline dönüştüren yenilik bu.

Şimdilik sınırlamayı anla: statik gömmeler gerekli ama yeterli değil. Bize kelimeler için matematiksel bir başlangıç noktası veriyorlar, ama bunları bağlamsal anlamlara birleştirmek için ek mekanizmaya ihtiyacımız var.

2.7 Uygulamalı Alıştırmalar

Teori iyi. Deneyim daha iyi. Gömmeleri pratikte görelim.

2.7.1 Keşfedilecek Çevrimiçi Araçlar

TensorFlow Embedding Projector (projector.tensorflow.org) - Kelime gömmelerini 3 boyutlu görselleştiriyor - Kelimeler arayabilir ve komşularını görebilirsin - Farklı kümeleri görmek için görselleştirmeyi döndürmeyi dene

Word2Vec Demo Siteleri - “word2vec online demo” ara - Birçok site kelime aritmetiği denemenize izin veriyor - “Kral - adam + kadın” yaz ve ne çıktığını gör

2.7.2 Alıştırmalar

Alıştırma 1: Kümeleri Bul

Bir gömme görselleştiricisinde şu kelimeleri ara ve yakınında ne kümelendiğini not et: - “mutlu”: Hangi duygular onunla kümelenecek? - “doktor”: Hangi meslekler komşu? - “kırmızı”: Hangi diğer kelimeler yakın?

Alıştırma 2: Kelime Aritmetiği Dene

Bir kelime aritmetiği demosu bulursan, şunları dene:

- Fransa - Paris + Londra = ?
- iyi - daha iyi + daha kötü = ?
- köpek - köpek yavrusu + yavru kedi = ?

Beklenen sonuçları aldın mı? Almadıysan, bunun nedeni ne olabilir?

Alıştırma 3: Farklı Olanları Bul

Bir gömme görselleştiricisinde “elma” ara. İki küme bulabilirsin:

1. Meyveler (muz, portakal, armut)
2. Teknoloji (iPhone, Mac, Google)

Aynı kelime farklı bağlamlarda görünüyor ve birden fazla konuma sahip olabiliyor (ya da anlamlar arasında bir uzlaşma olan bir konum).

2.8 Kontrol Noktası Alıştırması

Süre: 20-30 dakika **Malzemeler:** Kağıt veya elektronik tablo (kod gerekmez)

2.8.1 Talimatlar

10 hayvan için basit bir 3 boyutlu gömme sistemi oluştur.

1. **Hayvanları ayıran 3 özellik seç:**
 - Örnek: boyut, evcil, tehlikeli
 - Ya da: suda yaşayan, bacak sayısı, etçil
2. **Her hayvanı** her özellikte -1 ile +1 arasında puanla
3. **Gömmelerini oluştur:**

Hayvan	Boyut	Evcil	Tehlikeli
Kedi	-0.5	0.9	-0.7
Aslan	0.8	-1.0	0.9
Köpek	0.0	0.9	-0.3
Japon Balığı	-0.9	0.8	-1.0
Köpek Balığı	0.7	-1.0	0.9
Hamster	-0.9	0.9	-0.9
Ayı	0.9	-0.9	0.8

Hayvan	Boyut	Evcil	Tehlikeli
At	0.7	0.7	-0.2
Kurt	0.3	-0.8	0.6
Tavşan	-0.7	0.7	-0.9

4. Gömmelerini analiz et:

- Hangi hayvanlar en benzer? (3 boyutlu uzayında en yakın)
- “Kedi” ve “aslan” bir araya kümeleniyor mu? (kedigiller)
- “Kedi” ve “hamster” bir araya kümeleniyor mu? (küçük evcil hayvanlar)
- Hangi grupta daha güçlü ortaya çıkıyor?

5. Yansıma:

- Boyut seçiminin hangi benzerliklerin ortaya çıktığını belirledi mi?
- Hayvanları daha iyi ayırt etmek için hangi boyutları ekledin?
- 768 boyutun daha fazla nüans yakalayacağını görebiliyor musun?

2.9 Temel Çıkarımlar

Öğrendiklerin:

1. ASCII kodları yazımı yakalar, anlamı değil (“kedi” ve “kent” benzer görünür ama ilişkisiz)
2. Gömmeler, kelimeleri yüksek boyutlu bir anlamsal uzayda konumlar olarak temsil ediyor
3. Benzer kelimeler bir araya kümeleniyor; ilişkisiz kelimeler uzakta
4. Kelime aritmetiği işe yarıyor: kral - adam + kadın $\approx$ kraliçe
5. Bağlam anlamı öğretiyor: birlikte görünen kelimeler benzer gömmeler geliştiriyor
6. Statik gömmelerin bir sınırlaması var: ortalama alma kelime sırasını kaybediyor (“köpek adamı ısırdı” ile “adam köpeği ısırdı” aynı anlama gelmiyor)

Temel kavramlar:

- **Gömme (Embedding):** Bir kelimenin anlamsal uzaydaki konumunu temsil eden bir sayılar listesi (vektör)
- **Vektör/Boyut:** Daha fazla boyut (768+) daha nüanslı ilişkileri yakalıyor
- **Kosinüs benzerliği:** İki gömmenin ne kadar benzer olduğunu ölçüyor (1 = aynı, 0 = ilişkisiz, -1 = zıt)
- **Dağılımsal hipotez:** Benzer bağlamlarda görünen kelimelerin benzer anlamları var (“Bir kelimeyi arkadaşlarından tanırısın”)
- **Öz-denetimli öğrenme:** Veri yapısından öğrenme, insan etiketlerinden değil

Gömme hattı:

Kelime: "kral"

| gömme araması

v

Vektör: [0.82, 0.31, 0.91, -0.24, ...] (768 sayı)

| anlamsal uzay

v

"Kraliçe"ye yakın, "muz"dan uzak konum

i Gözden Geçirme Sorusu Yanıtları

Bu gözden geçirme sorularının tüm yanıtları **Ek D**'de mevcut.

2.10 Gözden Geçirme Soruları

1. Yapay zeka için kelimeleri temsil etmek üzere neden sadece ASCII kodlarını kullanamayız?
2. Gömme nedir ve neden böyle adlandırılıyor?
3. Modern gömmeler neden sadece 2 veya 3 yerine yüzlerce boyut kullanıyor?
4. “Kral - adam + kadın = kraliçe” denklemini kendi kelimelerin ile açıkla. Gömmeler hakkında ne ortaya koyuyor?
5. “Bir kelimeyi arkadaşlarından tanırsın” ne anlama geliyor? Gömme sistemleri bu fikri nasıl kullanıyor?
6. Cümleleri anlamak için kelime gömmelerinin ortalamasını almak neden yeterli değil? Bir örnek ver.

2.11 Sırada Ne Var

Artık bireysel kelimelerin anlam yakalayan sayılara nasıl dönüştüğünü anlıyoruz. Ama dil izole kelimelerden daha fazlası, ilişki içindeki kelimeler.

Şunu düşün: “Kedi yorgun olduğu için paspasın üzerinde oturdu.”

“O” neye atıfta bulunuyor? Kedi, açıkça. Ama bir bilgisayar bunu nasıl bilecek? “O” kelimesi birçok şeye atıfta bulunabilir. Anlamak, tüm cümleye bakmayı ve neyin neyle ilişkili olduğunu bulmayı gerektiriyor.

Bu **dikkat** sorunu: her kelimenin bağlamı anlamak için diğer kelimelere nasıl “bakmasına” izin veriyoruz? Dizileri ilişkileri koruyarak nasıl işleriz?

İşte Bölüm 3: Dikkat Mekanizması’nın konusu tam da bu. Modern BDM’leri mümkün kılan kilit yenilik.

Bölüm 3

Dikkat Mekanizması

“Her anımı, önündeki işi tam bir ciddiyetle, sevgiyle, istekle ve adaletle yapmaya odakla. Kendini tüm dikkat dağıtıcılardan kurtarmayı da unutma.” — **Marcus Aurelius**, *Düşünceler*

Ne Öğreneceksin - Eski yapay zeka yaklaşımları neden uzun metinlerle zorlandı - Yapay zeka bağlamında “dikkat” ne anlama geliyor - Sorgu (Query), Anahtar (Key) ve Değer (Value) birlikte nasıl çalışıyor - Çok başlı dikkat neden tek baştan daha güçlü - Her şeyi değiştiren 2017 makalesi

Temel Terimler

- *Dikkat (Attention)*: Kelimelerin diğer kelimelere seçici olarak odaklanmasını, hangilerinin ilgili olduğunu belirlemesini ve bilgiyi buna göre harmanlamasını sağlayan mekanizma [Sözlüğe bakın](#)
- *Öz-Dikkat (Self-Attention)*: Bir dizinin kendisine dikkat etmesi; her kelime aynı dizideki diğer tüm kelimelere bakabilir [Sözlüğe bakın](#)
- *Çapraz Dikkat (Cross-Attention)*: Bir dizinin farklı bir diziye dikkat etmesi; çeviri, altyazılama ve benzeri eşleştirmeli görevlerde kullanılır [Sözlüğe bakın](#)
- *Sorgu (Q), Anahtar (K), Değer (V)*: Gömmelerden türetilen üç vektör. Sorgu kelimenin ne aradığını, Anahtar kelimenin ne sunduğunu, Değer ise alınan gerçek bilgiyi temsil eder [Sözlüğe bakın](#)
- *Çok Başlı Dikkat (Multi-Head Attention)*: Paralel çalışan birden fazla dikkat hesaplaması; her biri farklı ilişki örüntüleri öğrenir [Sözlüğe bakın](#)
- *Softmax*: Skorları toplamı 1 olan olasılıklara dönüştüren fonksiyon; en yüksek skorlar en fazla olasılığı alır [Sözlüğe bakın](#)
- *Transformer*: “Attention Is All You Need” (2017) makalesindeki mimari; yalnızca dikkat mekanizmalarını kullanır, tekrarlama yok. Tüm modern BDM’lerin (Büyük Dil Modelleri) temelidir [Sözlüğe bakın](#)
- *Ölçeklendirilmiş Nokta Çarpım Dikkati (Scaled Dot-Product Attention)*: Nokta çarpımlarını $\sqrt{\text{boyut}}$ ’a bölen dikkat; Transformer’larda standart yaklaşım [Sözlüğe bakın](#)
- *Ağırlıklar (Weights)*: Dönüşümlerin nasıl çalıştığını tanımlayan, eğitimle öğrenilen sayılar. Gömmeyi ağırlıklarla çarpınca Q, K veya V elde ederiz [Sözlüğe bakın](#)

- *Dönüşüm (Transformation)*: Sayı kümesini ağırlıklarla çarparak başka bir kümeye dönüştürme işlemi. Gömmelerden Sorgu, Anahtar ve Değer oluşturmak için kullanılır [Sözlüğe bakın](#)

Kontrol Noktası

Bu bölümün sonunda şunları anlayacaksınız:

1. Dikkat mekanizmalarının nasıl çalıştığını ve yapay zekada neden devrim yarattığını
2. Sorgu, Anahtar ve Değer'in kelimelerin ilgili bağlamı bulmasını nasıl sağladığını
3. Çok başlı dikkatin farklı ilişkileri nasıl yakaladığını
4. 2017'deki Transformer makalesinin neden her şeyi değiştirdiğini
5. Öz-dikkat ile çapraz dikkat arasındaki farkı

Bölüm 2'de kelimeleri sayılara (anlamı yakalayan gömmelere) nasıl dönüştüreceğimizi öğrendik. Ama aynı zamanda bir duvara da çarptık: gömmelerin ortalamasını almak gibi basit yaklaşımlar önemli bilgileri kaybediyor. "Köpek insanı ısırdı" ile "İnsan köpeği ısırdı" aynı kelimelere ve aynı ortalama gömmeye sahip, ama anlamları tamamen farklı.

Kelime sırası önemli. Bağlam önemli. Uzak kelimeler arasındaki ilişkiler de önemli.

Bu bölümde çözümü tanıyacağız: **dikkat mekanizması**. Modern BDM'leri (Büyük Dil Modelleri) mümkün kılan ana yenilik bu ve onu anlamak, bundan sonra gelecek her şey için temel oluşturuyor.

3.1 Eski Yapay Zekanın Sorunu

Dikkatten önce yapay zekanın dil konusunda temel bir sorunu vardı: hatırlayamıyordu.

3.1.1 Sıralı Darboğaz

2017'den önce yaygın yaklaşım, Tekrarlayan Sinir Ağları (Recurrent Neural Networks, RNN) denilen bir şey kullanıyordu. Nasıl çalıştıklarına bakalım:

Bir cümleyi kelime kelime okuduğunu, ama sadece küçük bir yapışkan not kağıdına not alabildiğini düşün. Her kelimedenden sonra notunu güncelliyorsun, ama kağıt hep aynı boyutta kalıyor. Uzun bir paragrafın sonuna geldiğinde, başlangıçtaki bilgiler çoktan unutulmuş oluyor; yeni bilgiler eskilerini sıkıştırmış durumda.

Cümle: "Adamın sahip olduğu köpeğin kovaladığı kedi kaçtı"

RNN işleme:

Kelime 1: "Adamın"	→ [notlar: "adam"]
Kelime 2: "sahip"	→ [notlar: "adam sahip..."]
Kelime 3: "olduğu"	→ [notlar: "sahip olduğu..."]
Kelime 4: "köpeğin"	→ [notlar: "köpek..."]
Kelime 5: "kovaladığı"	→ [notlar: "kovaladı... ne kovaladı?"]
Kelime 6: "kedi"	→ [notlar: "kedi... bekle?"]
Kelime 7: "kaçtı"	→ [notlar: "kaçtı... ne kaçtı?"]

“kaçtı” kelimesine geldiğimizde neyin kaçtığını unutmuş oluyoruz. Kedi mi? Köpek mi? Adam mı? Model unutmuş. Teknik terimlerle buna **kaybolan gradyan sorunu (vanishing gradient problem)** deniyor: bilgi zincir boyunca ilerledikçe bozuluyor.

3.1.2 Uzun Metnin Gerektirdikleri

Şu cümleyi düşün:

“**Kupa bavula** sığmadı çünkü **o** çok büyüktü.”

“O” neye atıfta bulunuyor? Kupaya mı (kupa sığamayacak kadar büyük olduğu için) yoksa bavula mı (bavul çok büyük... dur bir dakika, bu mantıklı değil)?

Bu soruyu yanıtlamak için şunları yapman gerekiyor:

1. “O” kelimesine geldiğinde hem “kupa”yı hem de “bavul”u hatırla
2. Hangisinin mantıklı olduğunu anlamak için “çok büyük” ifadesini kullan
3. Cümlede birbirinden uzak olan kelimeleri bağla

Eski yapay zeka bu üçünün hepsinde zorlanıyordu. Herhangi bir anda herhangi bir kelimeye geri bakabilen ve neyin ilgili olduğunu anlayabilen bir şeye ihtiyacımız vardı.

Dikkate ihtiyacımız vardı.

3.2 Dikkat Nedir?

Dikkat, her kelimenin doğrudan diğer her kelimeye “bakmasına” izin vererek sıralı darboğazı çözüyor. Telefon oyununa gerek yok.

3.2.1 Kokteyl Partisi

Kalabalık bir kokteyl partisinde olduğunu hayal et. Onlarca konuşma etrafını sarıyor, uğultulu bir ses duvarı. Ama bir şekilde sadece tek bir konuşmaya odaklanabiliyorsun. Odanın karşısından biri adını söylediğinde hemen fark ediyorsun. Beynin gürültüyü filtreliyor ve önemli olana dikkat ediyor.

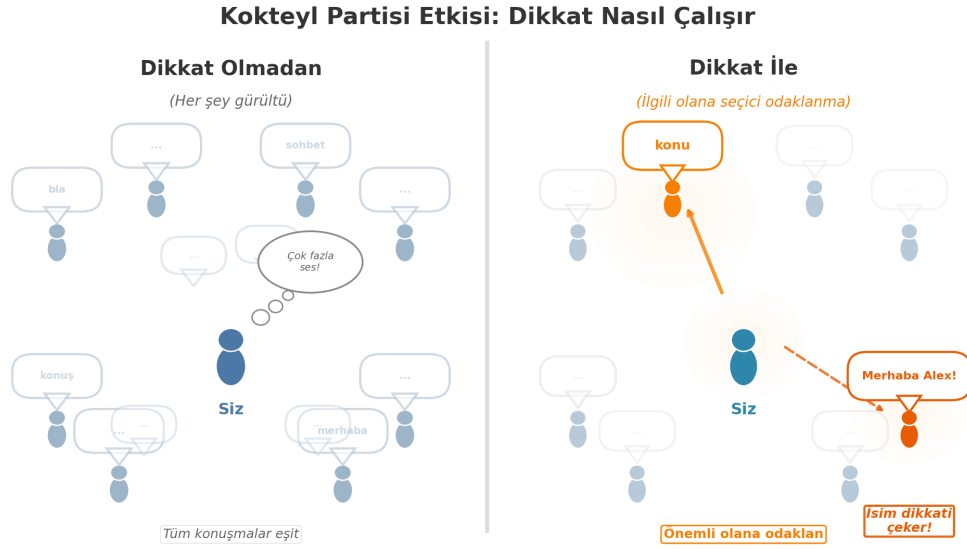
İşte dikkat tam olarak bu: **ilgiye dayalı seçici odaklanma**.

Şekil 3.1 bu karşılığı gösteriyor. Solda her ses eşit şekilde rekabet ediyor (saf gürültü). Sağda ise beynin önemli olanı vurgularken geri kalanını karartıyor. Soluk görünen konuşmaların tamamen görünmez olmadığına dikkat et: hala bir şekilde onların farkındasın, sadece daha az dikkat ediyorsun.

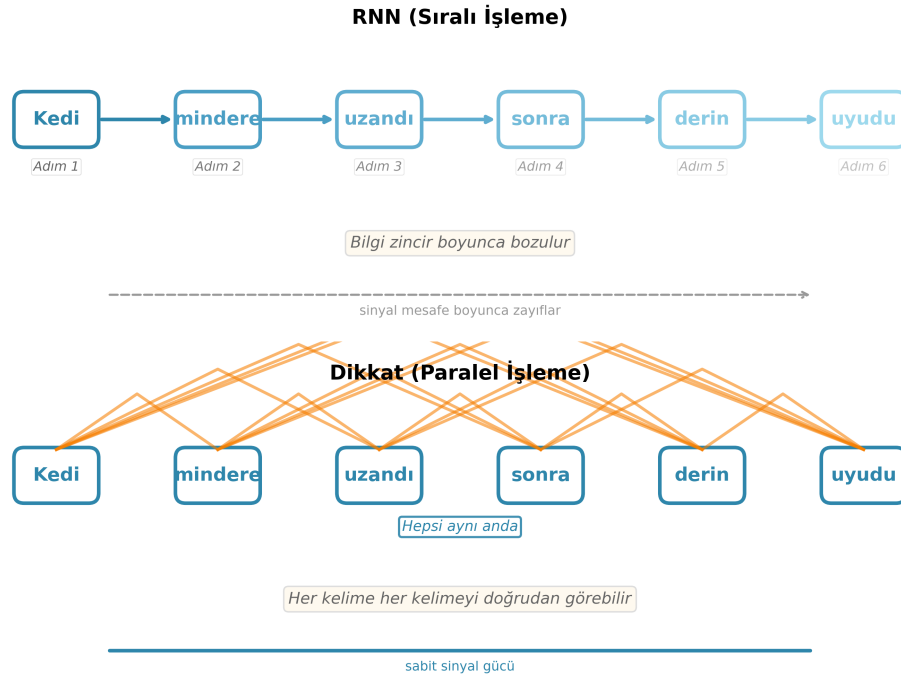
Yapay zekada dikkat benzer şekilde çalışıyor. Kelimeleri zincir boyunca birer birer işlemek yerine, her kelime doğrudan diğer her kelimeyi inceleyebiliyor ve anlamı için neyin ilgili olduğuna karar verebiliyor.

3.2.2 Sıralıdan Paralele

İşte temel değişim:



Şekil 3.1: Kokteyl Partisi Etkisi: Dikkat olmadan (solda), tüm konuşmalar eşit şekilde rekabet eder ve kafa karışıklığı yaratır. Dikkatle (sağda), kalabalık bir odada adını duymak gibi ilgili bilgilere seçici olarak odaklanılır.



Şekil 3.2: RNN kelimeleri bilginin solduğu bir zincir boyunca sırayla işler, Dikkat ise her kelimenin doğrudan diğer her kelimeye erişmesine izin verir.