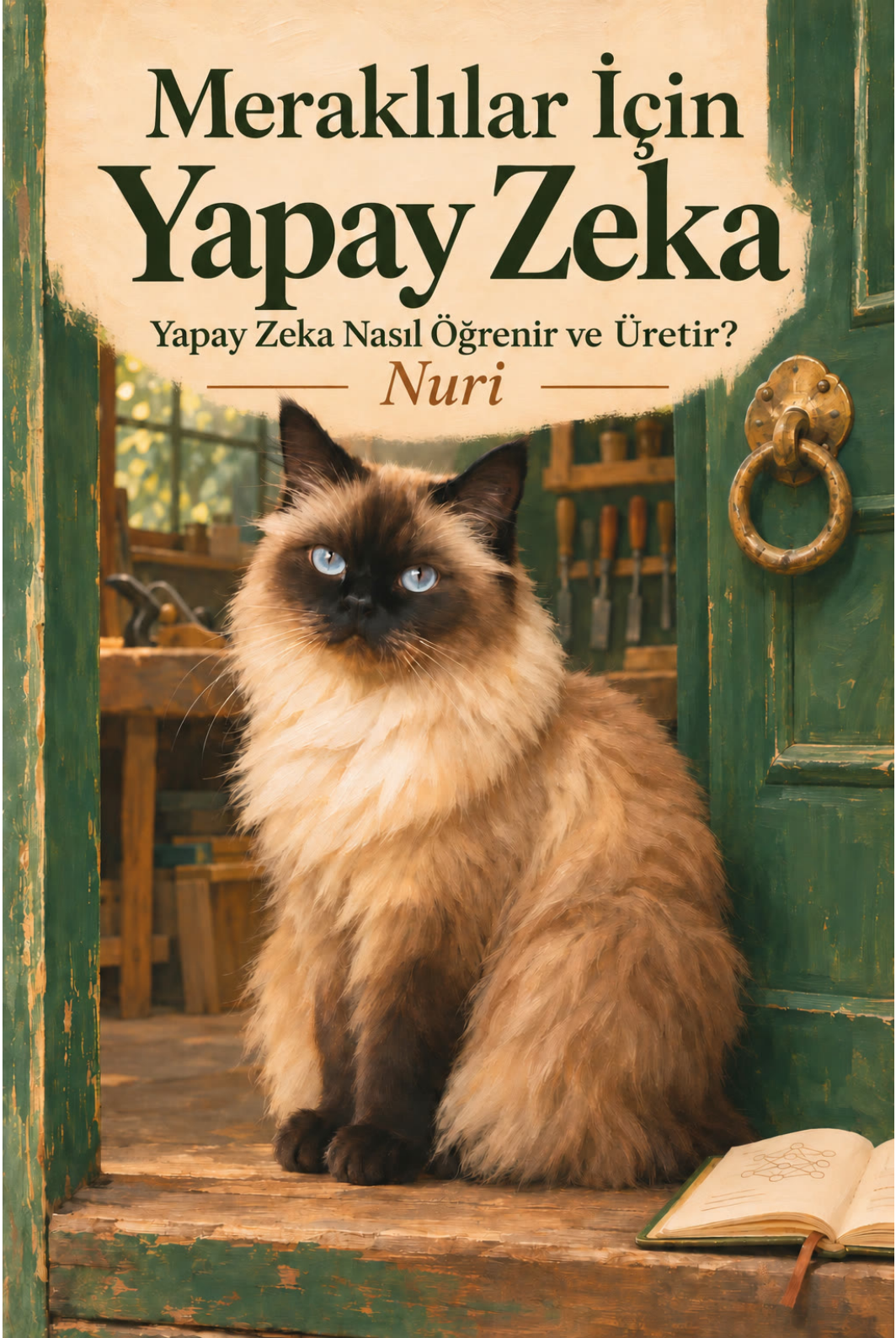
Meraklılar İçin Yapay Zeka

Yapay Zeka Nasıl Öğrenir ve Üretir?

Nuri



Meraklılar İçin Yapay Zeka

Yapay Zeka Nasıl Öğrenir ve Üretir?

Nuri

Sürüm 1.0 | 23 Kasım 2026

Bu kitap, sokak hayvanlarına ve onları besleyip şefkat gösteren kişilere adanmıştır.

Yayın Bilgileri

Meraklılar İçin Yapay Zeka
Yapay Zeka Nasıl Öğrenir ve Üretir?

Yazar: Nuri
İletişim: exforge@x386.org

Editoryal geliştirme ve içerik desteği: Charlie adıyla kullanılan OpenAI yapay zeka sistemi

Bu baskı hakkında

Bu ilk baskı kitabın 13 dersini, ortak giriş bölümlerini, genel değerlendirme ve kapanışı, ayrıca sözlüğü içerir. Okuyucu geri bildirimleri ve düzeltmeler sonraki sürümlerde değerlendirilecektir.

Herkes İçin Yapay Zeka, ilk kez bir yapay zeka asistanıyla çalışan okura nasıl soru soracağını, yanıtları nasıl kontrol edeceğini ve kişisel bilgilerini nasıl koruyacağını örneklerle anlatır.

Sürüm: 1.0
Tarih: 23 Kasım 2026

Lisans

Bu kitabın yazılı içeriği **CC BY-SA 4.0** lisansı altında sunulmaktadır. Bu lisans, ticari amaçlar dahil, kitabı kopyalamanıza, dağıtmanıza ve uyarlamanıza izin verir. Bunun için yazarı kaynak göstermeli, lisansa bağlantı vermeli, yaptığınız değişiklikleri belirtmeli ve uyarladığınız içeriği aynı lisansla paylaşmalısınız.

Lisansın özeti ve ayrıntıları

Bu kısa açıklama lisansın hukuki metninin yerini tutmaz.

Kitabın üretimi için geliştirilen özgün yazılımlar, ayrıca belirtilmediği sürece **GPL-3.0-or-later** lisansı altında yayımlanacaktır.

Üçüncü taraf yazı tipleri ve diğer varlıklar kendi lisanslarına tabidir.

İçindekiler

Giriş 1
Bu Kitap Kimler İçin Hazırlandı? 3
Bu Kitap Nasıl Kullanılmalı? 4
Öğrenme Yol Haritası 5
Bölüm 1 · Model eğitimi, parametreler ve geri yayılım 8
Bölüm 2 · Dil modellerinde sıradaki token tahmini ve çapraz entropi 15
Bölüm 3 · Token vektörleri, dikkat mekanizması ve softmax 20
Bölüm 4 · Sohbet modellerinde roller, bağlam ve cevap üretimi 26
Bölüm 5 · Araç çağrıları ve verinin güncelliği 31
Bölüm 6 · Belgelerde arama ve kaynağa dayalı cevap üretimi (RAG) 34
Bölüm 7 · Görüntü algılama, pikseller ve görsel özellikler 38
Bölüm 8 · Difüzyonla metinden görüntü üretimi 44
Bölüm 9 · Referans görüntüyle üretim ve maskeyle düzenleme 50
Bölüm 10 · Ses algılama, dalga biçimi ve spektrogram 54
Bölüm 11 · Konuşma tanıma ve metinden konuşma üretimi 59
Bölüm 12 · Videoda hareket algılama ve video üretimi 63
Bölüm 13 · Videoda ses ve görüntü uyumu, olay sırası ve nedensellik 68
Genel Değerlendirme ve Kapanış 72
Sözlük 74

Giriş

Telefonunuza “Kapı açık” dediğinizde ekranda bu iki kelimeyi görürsünüz. Aynı sözleri bu kez fısıldayın. Yazı yine aynı kalabilir, ama iki kaydı dinleyen biri aradaki farkı hemen duyar. Peki telefon, sesiniz bu kadar değiştiği hâlde aynı kelimeleri nasıl bulur? Bir de işlemi tersine çevirdiğimizi düşünün. Elimizde yalnızca “Kapı açık” yazısı varsa, bu sözleri seslendirecek bir sistem nerede duraklayacağına, hangi kelimeyi vurgulayacağına nasıl karar verir?

İlk kitabımız *Herkes İçin Yapay Zeka*’da gündelik hayatta karşılaştığımız sistemlerin hangi koşullarda işe yaradığını ve verdikleri cevaplara ne ölçüde güvenebileceğimizi ele almıştık. Bu kitapta o cevapların nasıl ortaya çıktığını araştıracağız. Bunun için bir ustayla çırağın çalıştığı küçük bir atölyeye uğrayacağız. İlk gün elimizde yalnızca bir sipariş defteri olacak. Sonra aynı atölyeden bir cümle, bir fotoğraf, kısa bir ses kaydı ve birkaç saniyelik bir video alıp her birini yakından inceleyeceğiz.

Defterde eski siparişlerin kaç parçadan oluştuğu ve ne kadar sürdüğü yazıyor. Yeni bir siparişin süresini tahmin etmek için her parçaya belli bir süre ayırdığımızı düşünün. Tahminimiz küçük siparişlerde de büyüklerde de beş dakika kısa çıkıyorsa, belki her işin başında yaptığımız hazırlığı hesaba katmamız gerekiyordur. İşte ilk **modelimiz** böyle bir hesap olacak. Hazırlığa ve her bir parçaya ayırdığı süreyi değiştirip eski siparişlerde ne kadar yanıldığına bakacağız. Bir modelin örneklerden **öğrenmesi**, bu hesapta ayarlanabilen sayıları, tahminlerdeki hatayı azaltacak biçimde değiştirmesi demek. Eğitimden sonra, süresini henüz bilmediğimiz bir sipariş için aynı hesabı kullanabileceğiz.

Bir dil modelinde de örneklerden öğrenilen sayılar vardır. Burada tahmin edilen şey, bir metnin nasıl devam edeceğidir. “Usta kapıyı…” diye başlayan bir cümle hem “açtı” hem de “kapattı” diye sürebilir. Öncesinde çırakların dışarıda beklediğini okuduysanız ilk seçenek size daha doğal gelebilir. Modelin önceki sözlerden nasıl yararlandığını ve seçtiği her yeni parçanın bir sonraki seçimi nasıl etkilediğini göreceğiz. Sonra çırağın telefonundan daha zor bir soru soracağız. “Atölyenin kapısı şu anda kilitli mi?” Ustanın sabah kapıyı kilitlediğini anlatan bir cümleyle kilitten az önce alınmış bir ölçüm, bu soruyu cevaplarken farklı değer taşır.

Atölyenin yeşil kapısını tarif ettiğimizde bir görüntü modeli pirinç tokmağıyla birlikte bir kapı resmi üretebilir. Oysa ustanın kendi

tokmağında küçük bir çentik vardır. O çentiğin de resimde görünmesini istediğimizde, modele tokmağın fotoğrafını vermek neyi değiştirir? Görüntüler üzerinde çalışırken bir fotoğrafın küçük renk noktalarıyla nasıl temsil edildiğini, modelin bunlar arasındaki ilişkilerden nasıl yararlandığını ve yeni bir resim üretirken hangi ayrıntıların değişebildiğini inceleyeceğiz.

Ses ve videoyla birlikte zaman da hesabımıza girecek. İki vuruş arasındaki sessizlik, tokmağın kaç kez vurulduğunu ayırt etmemize yardım eder. Aralık bir kapının tek fotoğrafına bakınca kapının açıldığını mı, kapandığını mı söylemek güçtür. Birkaç görüntüyü çekildikleri sırayla izleyince hareketin yönü ortaya çıkar. Kitap boyunca buna benzer küçük deneylerle hem bir modelin elindeki ipuçlarından neler çıkarabildiğine hem de yeni bir ses ya da video üretirken hangi ayrıntıları kendisinin tamamladığına bakacağız.

Bu Kitap Kimler İçin Hazırlandı?

Bu kitabı, bir yapay zeka modelinin verdiği cevabı okuyup “Bunu nasıl buldu?” diye soran okurlar için hazırladık. Kedinizi bir fotoğrafta, yüzünün bir bölümü gölgede kalsa bile tanıyabilirsiniz. Peki bir model, aynı fotoğraftaki açık ve koyu renkli noktalar arasında hangi ilişkileri öğrenirse bir kediyi başka bir hayvandan ayırt edebilir? Gündelik hayatta kolayca yaptığımız bir işin nasıl gerçekleştiğini merak etmek, burada ele alacağımız teknik sorular için iyi bir başlangıçtır.

Konuyla ilk kez tanışanlar kadar, yapay zeka araçlarını derslerinde veya mesleklerinde kullananlar da kitapta kendi deneyimlerine karşılık bulabilir. Örneğin bir görüşmenin ses kaydını yazıya döktüren bir araştırmacı için kelimelerin doğru aktarılması önemlidir. Ancak araştırdığı konu konuşanın tereddüdüyse, iki kelime arasındaki duraklama da anlam taşır. Böyle bir durumda hangi bilginin yazıda korunduğunu, hangisinin yalnızca ses kaydında kaldığını ayırt etmek gerekir. Kitap boyunca bir modelin ürettiği sonucu, o sonuca ulaşırken kullanabildiği bilgilerle birlikte değerlendireceğiz.

Programlama deneyimi ya da ileri matematik bilgisi gerekmiyor. Bir tablonun iki satırını karşılaştırabilmek, birkaç sayının ortalamasını alabilmek ve bir tahminin ölçülen değerden ne kadar saptığını görebilmek ilk örnekleri izlemek için yeterli. Matematiğe aşina olan okurlar, aynı örneklerde hesabın ayrıntılarını da inceleyebilir. Örneğin modelin iki ayarını aynı miktarda değiştirmenin tahmin hatasını neden farklı ölçülerde etkilediğini sayılarla takip edebilirler. Her teknik terimi ilk geçtiği örnek içinde açıklıyoruz. Kitabın sonundaki sözlük de ilerleyen bölümlerde bu kavramlara dönmeyi kolaylaştırıyor.

Bu Kitap Nasıl Kullanılmalı?

Yapay zekanın nasıl öğrendiğini ilk kez inceliyorsanız kitabı bölüm sırasıyla okumanızı öneririz. Ustanın sipariş defteriyle başlayan hikâyede her yeni soru, önceki bölümde öğrendiğimiz bir şeyi yeniden kullanmamıza fırsat verir. İlk bölümde bir süre tahminini ustanın ölçtüğü süreyle karşılaştırırız. İkinci bölümde aynı soruyu bir dil modeli için sorarız. Bir cümlenin birden fazla anlamlı devamı varsa modelin tahminini neyle karşılaştıracağız?

Atölyede bir soru sorulduğunda cevabı okumadan önce biraz durun. Örneğin on ve yirmi parçalık siparişlerin sürelerini gördüğünüzde otuz parça için kendi tahmininizi yazın. Yanına, bu sayıya nasıl ulaştığınızı anlatan bir cümle ekleyin. Usta yeni siparişin kaç dakika sürdüğünü deftere geçirdiğinde ilk notunuza dönün. Tahmininiz ölçümden farklıysa hangi varsayımınızı gözden geçirmeniz gerektiğini artık tartışabilirsiniz.

Bir tabloyu ya da formülü incelerken tek bir örneği baştan sona izlemek işinizi kolaylaştırır. Tablodan bir sipariş seçin. Önce kullanılan sayıların birimlerine bakın. Parça başına iki dakika ile sipariş başına beş dakika, hesaba nasıl giriyor? Sonra yalnızca parça sayısını değiştirin ve tahmini yeniden yapın. Matematiksel gösterime alışkın değilseniz ilk okumada bu ilişkiyi kendi sözlerinizle anlatabilmeniz yeterli. Hesabın ayrıntısına, hangi soruyu cevapladığını gördükten sonra dönebilirsiniz.

Küçük deneyleri bir yapay zeka uygulamasında da tekrarlayabilirsiniz. Örneğin ustanın kapıyı neden kilitlediğini önce hikâyenin tamamını vererek sorun. Ardından gerekçeyi açıklayan cümleyi çıkarın ve aynı soruyu yeni bir sohbette sorun. Yeni sohbet açmak, ilk metnin konuşma geçmişinden cevaba ipucu vermesini önler. İki cevabı karşılaştırırken her biri için “Bu açıklamayı destekleyen cümleyi modele verdiğim metinde gösterebiliyor muyum?” diye sorun.

Belirli bir konuya dönmek istediğinizde İçindekiler’deki bölüm adlarından yararlanabilirsiniz. Örneğin fotoğraftaki bir ayrıntının düzenleme sırasında neden değiştiğini merak ediyorsanız doğrudan dokuzuncu bölüme geçebilirsiniz. Bölüm başlığının altındaki italik soru sizi ilgili atölye sahnesine götürür. Bir kavramın kısa karşılığı için kitabın sonundaki Sözlük’e, konuların birbirine nasıl bağlandığını görmek için de Öğrenme Yol Haritası’na bakabilirsiniz.

Öğrenme Yol Haritası

Kitabın on üç bölümü dört aşamada ilerliyor. Önce bir modelin geçmiş örneklerden nasıl öğrendiğini inceleyeceğiz. Sonra öğrendiklerini kullanarak bir soruya nasıl cevap verdiğini ve bu cevabın hangi bilgilere dayandığını araştıracağız. Görüntüye, sese ve videoya geçtiğimizde aynı soruları yeniden sorabileceğiz. Bu kez elimizde bir cümlenin kelimeleri yerine bir fotoğrafın pikselleri ya da iki vuruş arasında geçen süre olacak.

1. Örneklerden öğrenme ve dilde tahmin · Bölüm 1–3

Bölüm 1 bir siparişin süresini tahmin etmekle başlar. Ölçülen süre ile tahmin arasındaki farkı hesaplar, modelin hangi ayarını değiştirmenin bu farkı nasıl etkilediğini izleriz. **Bölüm 2** aynı düşünceyi dile taşır. Bir cümlenin devamını tahmin eden model için eğitim örneği, önceden yazılmış metnin kendisidir. Modelin başarısını değerlendirirken yazarın kullandığı metin parçasına ne kadar olasılık verdiğine bakarız.

Bölüm 3 bu olasılığın nasıl hesaplandığını ele alır. Son kelimeleri aynı olan iki cümleye farklı başlangıçlar veririz. Modelin önceki kelimelerden aldığı bilgiyi nasıl birleştirdiğini izleyince, bir cümlenin başındaki küçük bir değişikliğin sonundaki tahmini neden etkileyebildiğini açıklayabiliriz.

2. Sohbet ve cevabın dayanakları · Bölüm 4–6

Bölüm 4 ile yarım bir cümleyi tamamlamaktan bir soruyu cevaplamaya geçeriz. Konuşmada kimin ne söylediğini ve sorudan önce hangi bilgilerin verildiğini inceleyerek modelin cevabını takip ederiz. **Bölüm 5** yeni bir güçlük getirir. Ustanın kapıyı öğlen kilitlediğini biliyoruz. Peki kapı akşam da kilitli mi? Akıllı kilitten gelen ölçümün zamanı, verebileceğimiz cevabı değiştirir.

Bölüm 6 geçmişteki bir olayı araştırır. Bu kez asistanın bakım defterinden hangi kaydı bulup modelin önüne getirdiğini izleriz. Bu üç bölümü tamamladığınızda bir cevaptaki iddiayı incelemek için

konuşmadaki hangi cümleye, hangi ölçüme ya da hangi tarihli kayda dönmeniz gerektiğini belirleyebilirsiniz.

3. Görüntüyü okuma, üretme ve düzenleme · Bölüm 7–9

Bölüm 7 kapının altından sızan ince bir ışık şeridiyle başlar. Birkaç pikselin parlaklığını karşılaştırarak görüntüdeki bir kenarın nasıl belirginleştiğini inceleriz. **Bölüm 8** bir adım öteye geçer. Ortada kapının fotoğrafı yokken, yazılı bir tarifin gürültüden başlayarak yeni bir görüntü üreten modele nasıl yol gösterdiğini görürüz.

Bölüm 9 için ustanın kendi tokmağının fotoğrafını da masaya koyarız. Fotoğrafı örnek göstermekle tokmağı fotoğraftan kesip kapının üzerine yerleştirmek farklı sonuçlar verir. Kapının yalnızca belirli bir yerini değiştirmek istediğimizde ise o alanı bir maskeyle işaretleriz. Bu aşamanın sonunda tokmağın üzerindeki küçük çentiğin hangi yöntemde niçin korunabildiğini tartışabiliriz.

4. Sesi, hareketi ve olay sırasını izleme · Bölüm 10–13

Bölüm 10 zaman içindeki değişimi ses üzerinden tanıtır. Kayda bakarak iki vuruşu birbirinden ayırır, ardından pirincin çınlamasını hangi ses özelliklerinin taşıdığını inceleriz. **Bölüm 11** bu ayrımı konuşmaya uygular. İki kişi aynı cümleyi söylediğinde ses kayıtları farklı olsa da yazıya aynı kelimeler geçebilir. O yazıyı yeniden seslendirirken bu kez vurgu ve söyleyiş için seçim yapmak gerekir.

Bölüm 12 hareketi anlamak için ardışık görüntüleri karşılaştırır. Aralık bir kapının hangi yöne gittiğini karelerin sırasından çıkarırız. Video üretirken de kapının görünüşünü ve hareketini kareler boyunca tutarlı kılmak gerekir. **Bölüm 13** ses ile görüntüyü aynı kayıtta buluşturur. Tokmağın vuruşunu duyup hemen ardından kapının açıldığını görebiliriz. Kapıyı açan kişi kadrajın dışında kaldığında, olayların sırasını bilsek bile o kişinin kimliği açık bir soru olarak kalır.

Yol haritasının sonunda

Kitabı bitirdiğinizde yeni bir örneği kendi başınıza incelemek için üç sorudan yararlanabilirsiniz. Model eğitim sırasında tahminini neyle karşılaştırmış olabilir? Şimdi elinde hangi bilgiler var?

Verdiği cevabın hangi ayrıntısını eldeki metinde, görüntüde ya da kayıtta gösterebilirsiniz? Ustanın defterinde başladığımız bu sorgulama biçimini kendi karşılaştığınız yapay zeka uygulamalarına da taşıyabilirsiniz.

Bölüm 1 · Model eğitimi, parametreler ve geri yayılım

Bir model öğrenirken içinde ne değişir?

Bir atölyeye otuz parçalık sipariş geldi. Müşteri işin ne kadar süreceğini soruyor. Usta defterini açıyor. Aynı parçadan on tane yaptıklarında iş 25 dakika sürmüş. Yirmi tane yaptıklarında ise 45 dakika.

Siz müşteriye kaç dakika söylerdiniz?

İkinci siparişte fazladan on parça için 20 dakika harcanmış. Buradan parça başına 2 dakika hesaplıyoruz. Ama ilk siparişte küçük bir bilmece var. On parçayı ikişer dakikadan saysak 20 dakika eder. Kalan 5 dakika nereye gitti? Belki usta her siparişin başında makineyi hazırlıyordur. Bu varsayımla otuz parça için cevabımız **5 + 2 × 30 = 65 dakika** olur.

Defterde hazırlık süresinin yazmadığına dikkat edin. İki kaydı açıklayabilecek bir ilişki kurduk. Şimdi bu ilişkinin biçimini bilgisayara verelim ve içindeki iki sayıyı ölçümlerden bulmasını sağlayalım.

Bir tahmini iyileştirmek

Modelimizin hesabı tek satır:

$$\text{Süre tahmini} = a + b \times \text{parça sayısı}$$

Burada a her siparişe bir kez eklenen süreyi, b ise her parça için eklenen süreyi gösteriyor. Eğitim sırasında değerlerini değiştirebildiğimiz bu sayılara **parametre** diyoruz. Hesabı rahat izlemek için başlangıçta $a = 0$ ve $b = 2$ seçelim. Hatayı da “tahmin eksi ölçüm” olarak tanımlayalım. Böylece eksi işaretini gördüğümüzde süreyi kısa tahmin ettiğimizi anlayacağız.

Parça sayısı	Ölçülen süre	İlk tahmin	Hata
10	25 dakika	20 dakika	−5 dakika
20	45 dakika	40 dakika	−5 dakika

İki tahmin de beşer dakika kısa. Tabloya bakan biri hemen “Hazırlık süresine beş dakika ekle, iş biter” diyebilir. Bilgisayara ise yüzlerce siparişte de kullanabileceği bir yöntem vermek istiyoruz. Parametreleri değiştirince bazı tahminler iyileşirken bazıları kötüleşebilir. Yeni ayarın genel olarak daha iyi olduğuna nasıl karar vereceğiz?

Bir yol, her hatanın karesini alıp ortalamasını hesaplamak. Kare almanın iki yararı var. Bir siparişi beş dakika erken, diğerini beş dakika geç tahmin edersek bu hatalar birbirini götürmez. Ayrıca büyük sapmalar daha çok ağırlık kazanır. Örneğin 2 dakikalık hata hesaba 4 olarak girerken 4 dakikalık hata 16 olarak girer.

Eğitim sırasında küçültmeye çalıştığımız bu ölçüye **kayıp** (*loss*) diyoruz. Başlangıç kaybımızı hesaplayalım:

$$\frac{(-5)^2 + (-5)^2}{2} = \frac{25 + 25}{2} = 25$$

Bunun birimi dakika karedir. İki siparişte de beşer dakika yanıldık. **25**, seçtiğimiz yöntemle bu iki hataya verdiğimiz puan. Parametreleri değiştirdikten sonra aynı hesabı yapıp puanın düşüp düşmediğine bakabiliriz.

Hangi ayarı hangi yönde değiştirelim?

Küçük bir deney yapalım. b yerinde dursun, a’yı 0’dan 0,01’e çıkaralım. Hazırlık süresi her siparişe bir kez eklendiği için iki tahmin de 0,01 dakika yükselir. Hatalar -5’ten $-4{,}99$’a gelir. Yeni kayıp $4{,}99^2$, yani $24{,}9001$. Demek ki başlangıç noktamızda a’yı biraz artırmak işe yarıyor.

Şimdi başa dönüp aynı deneyi yalnızca b ile yapalım. **Parça başına** 0,01 dakika eklersek on parçalık siparişin tahmini 0,10 dakika, yirmi parçalık siparişinki 0,20 dakika artar. Hatalar $-4{,}90$ ve $-4{,}80$ olur. Yeni kayıp da $(4{,}90^2 + 4{,}80^2)/2 = 23{,}525$ çıkar.

İki ayarı da aynı miktarda oynattık ama kayıp farklı ölçülerde değişti. Burada iki ayrı bilgi var. Birincisi **yön**: Ayarı artırınca kayıp büyüyor mu, küçülüyor mu? İkincisi **duyarlılık**: Küçük bir değişiklik kaybı ne kadar etkiliyor?

Bir ayarı tek başına değiştirdiğimizde kaybın o noktadaki değişim hızını **türev** verir. Her ayar için bulduğumuz türevi sırayla yazdığımızda **gradyanı** elde ederiz. Bizim iki ayarımız olduğu için gradyanda iki sayı var. Hesabın nasıl çıktığını merak ediyorsanız bir eğitim adımını birlikte izleyelim.

Bir eğitim adımının hesabı

Önce tek bir siparişe bakalım. Hata -5 iken tahmini küçük bir d miktarı kadar artırırsak yeni hata $-5+d$ olur. Karesini açalım:

$$(-5+d)^2 = 25 - 10d + d^2$$

Küçük bir adımda son terim çok küçük kalır. Örneğin $d = 0{,}01$ için ortadaki terim $-0{,}1$ iken son terim yalnızca 0,0001'dir. Başlangıçta tahmini biraz artırdığımızda karesel hatanın ne kadar değişeceğini yaklaşık olarak **−10 × artış** ile bulabiliriz. Buradaki -10, eğimdir.

İki siparişte de başlangıç hatası aynı. a'daki değişim iki tahmine de bire bir yansıdığı için kaybın a'ya göre eğimi $(-10-10)/2 = \mathbf{-10}$ olur.

b ise ilk tahmine 10, ikinciye 20 ile çarpılarak giriyor. Bu kez kaybın b'ye göre eğimi $[(-10)\times 10 + (-10)\times 20]/2 = \mathbf{-150}$ olur.

Gradyanımız **(−10, −150)**. Bu iki sayı bulunduğumuz noktadaki eğimleri veriyor. Atacağımız adımın büyüklüğünü belirlemek için bir katsayı seçiyoruz. Adı **öğrenme oranı** (*learning rate*). Örneğimizde 0,001 olsun. Her parametreden kendi eğiminin 0,001 katını çıkaralım:

$$a_{\text{yeni}} = 0 - 0{,}001 \times (-10) = 0{,}01$$

$$b_{\text{yeni}} = 2 - 0{,}001 \times (-150) = 2{,}15$$

Eğimlerin ikisi de eksiydi. Eksi bir sayıyı çıkardığımız için iki ayar da arttı. Yeni tahminler 10 parça için **21,51**, 20 parça için **43,01 dakika**. Kayıp da yaklaşık **8,07** oldu.

Bir eğitim adımının ardından bilgisayarda ne değişti? Aynı formül ve aynı siparişler duruyor. Formülün içinde sakladığımız iki sayı değişti. Bu yüzden bir sonraki hesapta farklı tahminler alıyoruz.

Burada şaşırtıcı bir ayrıntı var. Defterden parça başına 2 dakika hesaplamıştık. Algoritma tam o değerden başlayıp 2,15'e gitti. Nasıl oldu da kayıp düştü?

Hazırlık süresini başlangıçta sıfır aldığımız için iki tahmin de kısaydı. O noktada parça başına süreyi artırmak açığın bir kısmını kapattı. Algoritma sonraki adımda eğimleri yeni yerinde tekrar hesaplayacak. Bir ayarın hangi yönde ilerlemesi gerektiği, diğer ayarın o anki

değerine de bağlı. Öğrenme oranıyla adımın büyüklüğünü sınırlamamızın bir sebebi bu. Bulunduğumuz yerde aşağıyı gösteren eğim, aynı yönde istediğimiz kadar ilerleyebileceğimizi söylemez.

Otuz parçalık sipariş bitince

Usta defterine yeni bir satır ekliyor. Otuz parçalık sipariş **95 dakika** sürmüş. İlk iki kayda tam uyan 65 dakikalık tahminimiz yarım saat kısa kaldı. Peki nerede yanıldık?

Atölyeye gidince sebebini öğreniyoruz. Makine ilk yirmi parçayı teslimata hazır çıkarıyor. Aynı siparişte yirmiyi aşan her parça için ayrıca üç dakikalık el işlemi gerekiyor. Otuz parçanın on tanesi için fazladan 30 dakika harcanmış.

İlk iki siparişten bu kuralı çıkarmak mümkün müydü? İkisinde de el işlemi gerekmemişti. Üstelik formülümüz, *her yeni parçanın süreye aynı miktarı eklediği* varsayımına dayanıyordu. Hangi a ve b değerlerini seçersek seçelim, tek bir doğruyla bu üç kaydın üçünü birden tutturamayız. Yeni sipariş, eğitim kayıtlarına bakarak göremediğimiz bir eksikliği ortaya çıkardı. Bir modelin öğrendiklerini yeni örneklere taşıyabilmesine **genelleme** diyoruz.

Hesabı düzeltmek için önce şu soruyu cevaplayalım: **Kaç parça el işlemine girecek?**

Sipariş	Parça sayısından 20 çıkarınca	El işlemine girecek parça	Eklenecek süre
18 parça	−2	0	0 dakika
20 parça	0	0	0 dakika
23 parça	3	3	9 dakika

Ortadaki iki sütun arasında küçük bir işlem var. Negatif iki parçaya el işlemi yapamayacağımıza göre −2 yerine sıfır alıyoruz. Sonuç pozitif çıkarsa o sayıyı olduğu gibi kullanıyoruz. Bunu tek satırda “**sıfırla karşılaştır, büyük olanı al**” diye yazabiliriz.

Aktivasyon fonksiyonunu kim seçiyor?

Az önce uyguladığımız işlemin sinir ağlarında da bir adı var: **ReLU**. Formülü kısa:

$$\mathrm{ReLU}(z) = \max(0, z)$$

Buradaki z, fonksiyonun eline geçen sayı. Bizim örneğimizde $z =$ parça sayısı $- 20$. Dolayısıyla ReLU'ya 18 parçalık siparişte -2, 23 parçalıkta 3 giriyor. Fonksiyondan çıkan değerler ise sırasıyla 0 ve 3. Sonraki işlem bu çıktıyı üç dakikayla çarpıyor.

Bir sinir ağındaki birim de benzer biçimde çalışabilir. Önce girdilerini ağırlıklarla çarpıp toplar ve bir sabit ekler. Ardından bulduğu ara sayıyı bir **aktivasyon fonksiyonundan** geçirir. ReLU bu amaçla kullanılan fonksiyonlardan biridir. "Aktivasyon" sözcüğü bir açma kapama düğmesini çağrıştırabilir. Oysa tabloda gördüğümüz gibi çıktı, etkinin miktarını da taşıyor. Üç fazla parça ile on fazla parça sonraki hesaba farklı değerler gönderiyor.

Peki model eğitim sırasında ReLU'yu kendisi mi buldu? **Yaygın uygulamada aktivasyon fonksiyonunun türünü modeli tasarlayanlar seçer. Standart ReLU'nun kuralı eğitim boyunca sabittir.** Öğrenme, bu fonksiyonun çevresindeki sayılarda gerçekleşebilir. Atölye hesabında üç ayrı kararın yerini görebiliyoruz:

- **ReLU'nun yaptığı işlem** eline gelen sayıyla sıfırın büyüğünü almak.
- **20 sayısı**, parçaları sayarken kullandığımız eşik. ReLU'nun öncesinde, parça sayısından çıkarılıyor.
- **3 dakika**, eşik aşıldıktan sonra her fazla parçanın süreye katkısı. ReLU'nun çıktısıyla çarpılıyor.

Biz eşiği ve ek süreyi ustadan duyduk. Elimizde yalnızca sipariş kayıtları olsaydı, 20 ve 3 yerine öğrenilebilir iki parametre koyabilirdik. Yeterli sayıda, özellikle de eşiğin iki yanından örnekle eğitim sırasında bu değerleri ayarlamak mümkün olurdu. Eşik değişince grafiğin kırıldığı yer, parça başına ek süre değişince de kırılmadan sonraki eğim değişirdi. Kendi içinde öğrenilebilir parametreler taşıyan başka aktivasyon türleri de vardır.

Bu küçük işlemin ne kazandırdığını görmek için iki komşu siparişi karşılaştıralım. 19 parçadan 20'ye çıkınca süre 2 dakika uzuyor. 20'den 21'e çıkınca ise 2 dakika üretim ve 3 dakika el işçiliğiyle toplam 5 dakika uzuyor. **Aynı "bir parça daha" değişikliği, bulunduğumuz noktaya göre farklı sonuç verebiliyor.** Atölyedeki doğrusal olmayan ilişkiyi artık ifade edebiliyoruz.

Sonucu biliyoruz. İlk katman nasıl öğrenecek?

Atölyede yalnızca toplam süreyi ölçtüğümüzü düşünelim. Elimizde hazırlık, üretim ve el işçiliği için ayrı ayrı tutulmuş süreler yok. Modelin en sonunda yanıldığını görebiliyoruz. Peki hesabın başındaki bir ağırlığın bu yanılmaya etkisini nasıl bulacağız?

Bir an için hesabı iki çarpıma indirelim. Girdimiz 3. İlk katman bunu kendi ağırlığı olan 2 ile çarpıp **6** üretiyor. İkinci katman bu 6'yı kendi ağırlığı olan 4 ile çarpıp **24** üretiyor. Beklediğimiz cevap 20. Dört birim fazla tahmin ettik.

Şimdi ilk ağırlığı 2'den 2,001'e çıkardığınızı düşünün. Daha hesabı yapmadan, son tahminin hangi yöne gideceğini söyleyebilir misiniz? İlk katmanın çıktısı biraz artacak. İkinci katman bu artışı dörtle çarpacak. Zaten yüksek olan tahmin biraz daha yükselecek.

Rakamla izleyelim. Ağırlığı 0,001 artırınca ara değer 6'dan 6,003'e çıkar. Son tahmin de 24'ten 24,012'ye çıkar. İlk ağırlıktaki küçük değişikliğin etkisini iki işlem boyunca takip edebildik.

Şimdi aynı hesabı öteki uçtan okuyalım. **Son tahmin fazla. Bu tahmini hangi ara değer etkiledi?** İkinci katmana gelen sayı. **Ne ölçüde etkiledi?** İkinci katman o sayıyı dörtle çarptı. **O ara değeri hangi ağırlık etkiledi?** İlk katmandaki ağırlık. Onun etkisini de girdi olan 3 belirledi. **Geri yayılım** (*backpropagation*), kayıptan başlayıp bu yerel etkileri sondan başa birleştirerek her ağırlığın gradyanını hesaplar. Ardından eğitim algoritması bu gradyanları kullanıp ağırlıkları günceller.

İlk katmana hiçbir aşamada "Senin üretmen gereken ara sayı şuydu" demedik. Son hedefi ve aradaki işlemleri bilmek, o ağırlığı biraz değiştirmenin kaybı nasıl etkileyeceğini hesaplamamıza yetti.

Bu örnekte önce 2, sonra 4 ile çarpmayı tek bir "8 ile çarp" işleminde birleştirebilirdik. Araya atölyedeki ReLU gibi doğrusal olmayan işlemler girdiğinde ise katmanlar daha farklı ilişkiler kurabilir. Bir katmanın çıkardığı özellik, diğerinin kullanacağı girdiye dönüşür. Bunun neden önemli olduğunu görmek için atölyede başka bir işe bakalım.

Ustanın defterinden bir bisiklet fotoğrafına

Usta bu kez üretim süresini tahmin etmek yerine **"Bu parça kalite kontrolünden geçer mi?"** sorusuna cevap arıyor. Geçmişte her parçanın üzerindeki en uzun çiziği ve kenarındaki çapak miktarını ölçüp kaydetmiş olsun. Bu ölçülerle eski kontrol sonuçlarını yan

yana koyarak tahmin yapmayı deneyebiliriz. Artık cevabımız dakika cinsinden bir sayı yerine iki olasılıktan biri: geçer veya kalır. Süre tahmininde saatin gördüğü işi burada ustanın kaydettiği kontrol sonucu görüyor.

Şimdi de usta ölçümleri bıraksın, yalnızca **parçanın fotoğrafını ve kontrol sonucunu** kaydetsin. Sorduğumuz soru aynı. Ama çizik uzunluğunu artık kimse modele hazır vermiyor. Fotoğrafta bu bilgiyi taşıyan bir şey var mı? Belki düzgün bir yüzey boyunca uzanan ince, koyu bir çizgi. Modelin, piksellerden yola çıkarak karar vermeye yarayacak özellikleri de öğrenmesi gerekiyor.

Bir fotoğrafta bisiklet tanımaya buradan geçebiliriz. Elimizde bisiklet içeren ve içermeyen fotoğraflar olsun. Her fotoğraf için yalnızca "var" ya da "yok" etiketi verilmiş. Tekerleklerin yerini işaretlemedik, gidonun şeklini tarif etmedik.

Bir görüntü ağının ilk katmanlarında belirli yönlerdeki açık-koyu geçişlerine duyarlı filtreler gelişebilir. Sonraki katmanlar bu geçişlerin görüntüde nasıl bir araya geldiğini kullanabilir. Model bisiklet olan bir fotoğrafa düşük bir "bisiklet var" olasılığı verirse, etiketi bildiğimiz için bir kayıp hesaplayabiliriz. Geri yayılım sayesinde bu kayıp, görüntüyü ilk işleyen filtrelerin ayarlanmasına kadar etki eder.

Jantın yerini tek tek göstermeden bisiklet tanımayı öğretebilmemizin yolu böyle açılır. Atölyede ara değere "el işlemine girecek parça sayısı" adını biz vermiştik. Görüntü ağında öğrenilen bir ara özelliğe aynı kesinlikle "jant" adını vermek her zaman mümkün olmaz. Yine de o özelliğin nasıl öğrenilebildiğini anlayabiliyoruz. Eğitim sırasında bir fotoğrafın sonunda ölçtüğümüz kayıp, hesabın en başındaki ağırlıkların da ayarlanmasına yardımcı oluyor.

Atölyede elimizde bir saat, bisiklet fotoğrafında bir etiket vardı. Şimdi bir dil modelinin eğitim metninde şu cümleyi okuduğunu düşünelim: "Usta sabah erkenden atölyenin kapısını…" Burada birden fazla anlamlı devam var. **Öyleyse bir dil modeli öğrenirken tahminini neyle karşılaştırıyor?**

Bölüm 2 · Dil modellerinde sıradaki token tahmini ve çapraz entropi

Bir dil modeli neyi tahmin eder, hatasını nasıl ölçer?

“Usta sabah erkenden atölyenin kapısını…”

Cümleyi siz tamamlasaydınız hangi kelimeyi seçerdiniz? Sabah erkenden işinin başına gelen bir ustayı gözünüzde canlandırdıysanız aklınıza “açtı” gelmiş olabilir. Bir de hikâyeyi yazan kişinin elindeki metne bakalım:

> Usta sabah erkenden atölyenin kapısını kilitledi. O gün çalışacağı tezgâh müşterinin fabrikasındaydı.

İkinci cümleyi okuyunca ustanın sabah sabah kapıyı neden kilitlediğini anlıyoruz. Oysa “kapısını” kelimesinde durduğumuzda bu bilgi elimizde yoktu. “Açtı” o anda gayet makul bir tahmindi. Peki bu satırı eğitimde kullansaydık dil modelinin tahminini nasıl değerlendirirdik?

Önceki bölümde usta saate bakmış, deftere 25 dakika yazmıştı. Biz de tahmini bu ölçümle karşılaştırmıştık. **Bir sonraki metin parçasını tahmin ederek eğitilen bir dil modelinde hedefi metnin kendisi verir.** Bizim örneğimizde yazar “kilitledi” yazmış. Eğitimde soracağımız soru şu: Model, o ana kadar okuyabildiği metne dayanarak bu devama ne kadar olasılık vermişti?

Bir cümleden kaç soru çıkar?

Diyelim ki modelin sözlüğüne “atölye”, “atölyede”, “atölyeden” ve “atölyenin” için ayrı ayrı birer kelime yazıyoruz. Peki yarın “atölyelerimizden” karşımıza çıkarsa? Metni daha küçük parçalarla temsil edebilirsek elimizdeki parçaları yeni kelimelerde de kullanabiliriz.

Bu parçaların her birine *token*, metni belirli bir sözlük ve kurallar yardımıyla parçalara ayırıp her parçayı sözlükteki numarasına dönüştüren bileşene *tokenizer* denir. Örneğin küçük bir sözlük kurup “atölye”ye 17, “nin”e 42 numarasını verelim. Kurallarımız “atölyenin” kelimesini bu iki parçaya ayırıyorsa modelin girişinde sırasıyla 17 ve 42 yer alır. Bu numaralar sözlükte birer adres görevi görür. Başka

bir tokenizer aynı kelimeyi tek parça alabilir veya başka yerlerinden bölebilir. Parça sınırlarının Türkçedeki kök ve ek sınırlarına denk gelmesi gerekmez.

Bu ayrım tahmini de etkiler. Bizim küçük sözlüğümüzde model 17'nin ardından 42'yi seçerse, okurun henüz tek kelime olarak gördüğü "atölyenin" iki tahmin adımında yazılmış olur.

Kâğıt üzerinde hesabı izleyebilmek için küçük bir varsayım yapalım. Bizim örneğimizde her kelime bir token, nokta da ayrı bir token olsun. Şimdi cümleyi bir kâğıda yazın ve sağ tarafını elinizle kapatın. Elinizi her seferinde bir kelime sağa kaydırdığınızda aynı satırdan yeni bir soru çıkar.

Görülen bağlam	Sıradaki token
Usta	sabah
Usta sabah	erkenden
Usta sabah erkenden	atölyenin
Usta sabah erkenden atölyenin	kapısını
Usta sabah erkenden atölyenin kapısını	kilitledi
Usta sabah erkenden atölyenin kapısını kilitledi	.

Her satırın soluna **bağlam**, sağına **hedef** diyebiliriz. Bir satırda hedef olan "kapısını", sonraki satırda bağlamın parçasıdır. Eğitim yazılımı her ikisini de aynı metinden çıkarabilir. Kimsenin oturup her cümle için ayrı ayrı "soru" ve "doğru cevap" hazırlaması gerekmez. Hedefin bu şekilde verinin içinden elde edilmesine **özdenetimli öğrenme** (*self-supervised learning*) denir.

Peki bütün cümle eğitim yazılımının elindeyse model cevabı da görmüş olmuyor mu? Tablodaki sınır bunun için önemli. Hesaplar birlikte yapılsa bile her konumdaki tahmin yalnızca o konuma kadarki parçaları kullanacak biçimde düzenlenir. "Kapısını"ndan sonrasını tahmin eden konum, yazarın "kilitledi" seçimini henüz göremez. Eğitim yazılımı ise tahmini değerlendirmek için o seçimi bilir.

İki model de "açtı" derse ikisi de aynı ölçüde mi yanılır?

Modelin bu noktada ürettiği şey, sözlüğündeki token'lar üzerinde bir **olasılık dağılımı**. Bir an için iki farklı modelin aynı yarım cümleyi okuduğunu düşünelim. Aşağıdaki yüzdeleri örnek için biz seçtik.

Sıradaki token	A modeli	B modeli
açtı	%60	%50
kilitledi	%10	%40
Diğer bütün token'ların toplamı	%30	%10
Toplam	**%100**	**%100**

İkisinden de en olası tek kelimeyi isteseydik ikisi de "açtı" diyecekti. Yalnızca "bildi mi, bilemedi mi?" diye baksak aralarında fark göremezdik. Oysa B, ustanın kapıyı kilitlemesi ihtimaline ciddi bir pay ayırmış. A ise aynı devamı oldukça düşük olasılıklı bulmuş.

Şimdi elimizi kâğıttan çekelim. Yazar "kilitledi" yazdığına göre bu adımda tablonun **kilitledi** satırını okuyacağız. A'nın verdiği olasılık %10, B'ninki %40. **Bu örnekte B'nin kaybı daha düşük olmalı.** Bunu hesaplamak için iki modelden de bir kelime seçmesini istememize gerek kalmadı. Hedefi biliyoruz, dağılımdaki yerini bulup ona verilen olasılığı okuyabiliyoruz.

Küçük bir deneme yapabilirsiniz. Yazarın son kelimesini "açtı" olarak değiştirin ama modellerin yüzdelerine dokunmayın. Bu kez hangi model daha düşük kayıp alır? Cevap A, çünkü bakacağımız satır değişti. Tek bir cümle bize o örnekte hangi tahminin daha iyi puan aldığını söyler. Modelleri daha genel olarak karşılaştırmak için aynı hesabı çok sayıda konumda ve eğitimde kullanılmamış metinlerde yapmak gerekir.

Yüzdeyi kayba nasıl çeviriyoruz?

İlk bölümde süre tahmininin hatasını kareye almıştık. Burada elimizde başka türden bir sayı var: hedef token'a verilen olasılık. Hedefe yüksek olasılık verildiğinde küçük, çok düşük olasılık verildiğinde büyük bir kayıp istiyoruz. Bunun için yaygın olarak kullanılan hesabı aşağıdaki kutuda görebilirsiniz.

İki sayıyla çapraz entropi

Hedef token'a verilen olasılığa p diyelim. Bu konum için **çapraz entropi kaybı** $-\ln(p)$ olur. Logaritma, bir sayıya ulaşmak için tabanı hangi üsse yükseltmemiz gerektiğini söyler. Örneğin $10^{-2} = 0{,}01$ olduğu için 0,01'in 10 tabanındaki logaritması -2'dir. Buradaki ln aynı soruyu e (yaklaşık 2,718) tabanında sorar. 0 ile 1 arasındaki olasılıklar için üs negatiftir. Formülün başındaki eksi işareti kaybı pozitif yapar. Hesap makinesine yüzdeyi ondalık sayı olarak giriyoruz.

- A modeli için $p = 0{,}10$. Kayıp yaklaşık **2,30**.
- B modeli için $p = 0{,}40$. Kayıp yaklaşık **0,92**.

> Ölçeğin bir ucunu görmek kolay: Hedefe 1 olasılık verilirse $\ln(1) = 0$ olduğu için kayıp sıfır olur. Öteki yönde ise hedefe verilen olasılık sıfıra yaklaştıkça kayıp büyür. Yazarın kullandığı kelimeye neredeyse hiç şans tanımamanın bedeli yüksektir.

Logaritmanın burada ikinci bir yararı var. Diyelim ki model, metinde gerçekten geçen “kilitledi”ye %40, ardından “kilitledi”yi görünce noktaya %50 olasılık verdi. Bu iki parçanın art arda gelmesine verdiği olasılık $0{,}40 \times 0{,}50 = 0{,}20$ olur. Logaritma çarpımı toplama dönüştürdüğü için iki parçanın toplam kaybını ayrı ayrı hesaplayıp toplayabiliriz: $0{,}92 + 0{,}69 \approx 1{,}61$. Doğrudan $-\ln(0{,}20)$ hesaplasak da yuvarlandığında aynı sayıyı buluruz.

Burada “B’nin kaybı 0,92” derken modelin yüzde 92 yanıldığını söylemiyoruz. Bu sayı, iki tahmini aynı ölçekte değerlendirmemizi sağlayan bir puan. Eğitimde çok sayıda hedef token için bu puanlar hesaplanır ve genellikle ortalaması alınır. Ardından ilk bölümde tanıştığımız mekanizma çalışır. Geri yayılım, kaybın ağırlıklara göre gradyanlarını hesaplar. Eğitim algoritması da ağırlıkları günceller.

Bu durumda model her sabah hikâyesini “kilitledi” diye bitirmeyi mi öğrenecek? Başka bir yazar tam aynı başlangıçtan sonra “açtı” yazmış olabilir. O örnekte de “açtı”ya verilen olasılık kaybı belirler. Eğitim sırasında aynı ağırlıklar pek çok örnek için kullanılır. “Sabah erkenden” gibi bir ifade bir ipucu taşırken, daha önceki bir cümlede geçen “Bugün müşteriye gidiyorum” sözü farklı bir ipucu taşıyabilir. Farklı devamların olasılık kazanması, dilde aynı bağlamın birden fazla anlamlı devamı olabilmesini hesaba katar.

Bu kez devamını biz yazdırırsak

Şimdi eğitim metnini kapatalım. Eğitilmiş modele yalnızca “Usta sabah erkenden atölyenin kapısını” yazıp devam etmesini istediğimizi düşünelim. Artık elimizde bir yazarın birazdan açıp bakacağımız devam cümlesi yok. Model yine bir sonraki token için olasılıklar üretiyor. Bu kez dağılımı, **yazılacak parçayı seçmek** için kullanıyoruz.

A modelinin tablosuna dönelim. Her adımda en yüksek olasılıklı token’ı seçen bir yöntem kullanırsak ilk seçim “açtı” olur. Olasılıklara göre örnekleme yaparsak “kilitledi” de seçilebilir. Diyelim ki “açtı” seçildi. Bir sonraki adımda modelin önündeki metin artık şöyle:

> Usta sabah erkenden atölyenin kapısını açtı

Şimdi sırada ne olabilir? Bir nokta gelebilir. “Ve” gelirse usta atölyeye girip bir sonraki işine geçebilir. Eski tablodaki ikinci sıraya bakmak bize cevabı vermez. O tablo, sonu “kapısını” ile biten bağlam içindi. **Seçilen token bağlama eklenir ve yeni bağlam için olasılıklar yeniden hesaplanır.** Metin böyle uzar.

Aynı cümleyi bir modelin yaşamındaki üç ayrı anda düşünelim. **Sıfırdan eğitime başlamadan önce** tokenizer cümleyi parçalarına ayırabilir ve model bir olasılık dağılımı hesaplayabilir. Ama başlangıç ağırlıkları henüz Türkçe metinlerde “kapısını”dan sonra hangi parçaların geldiğinden bir şey öğrenmemiştir. **Eğitim sırasında** yazılmış devamı biliyoruz. “Kilitledi”ye verilen olasılıktan kaybı hesaplıyor, birçok örnekten gelen gradyanlarla ağırlıkları güncelliyoruz. **Eğitimden sonra**, aynı yarım cümleyi tamamlatırken bu kez öğrenilmiş ağırlıkları kullanıyoruz. Model “açtı” yazdığında bir sonraki tahmini değişir, çünkü artık bağlamda “açtı” da vardır. Bu bir kelimelik ilerleme için ağırlık güncellemesi gerekmez.

Artık tablodaki yüzdeleri nasıl değerlendireceğimizi ve onlarla nasıl metin üreteceğimizi biliyoruz. Ama yüzdeleri tabloya şimdilik biz yazdık. Gerçek bir modelin önüne bu cümle geldiğinde **“sabah erkenden” ile “kapısını” arasındaki ilişki hangi hesaplardan geçip “açtı” için bir olasılığa dönüşüyor?**

Bölüm 3 · Token vektörleri, dikkat mekanizması ve softmax

Aynı kelimelerle biten iki cümle neden farklı devam eder?

> **A.** Çıraklar dışarıda bekliyordu. Usta atölyenin kapısını…
>
> **B.** Herkes çıkmış, içerideki ışıklar sönmüştü. Usta atölyenin kapısını…

İlk metinde usta kapıyı açsa, ikincisinde kilitlese çoğumuz hikâyeyi yadırgamadan okumaya devam ederiz. Şimdi iki metnin de ilk cümlesini elinizle kapatın. Elinizde kalan kelimelere bakarak hangisinde "kilitledi" diyeceğinizi seçebilir misiniz?

Önceki bölümde "açtı" ve "kilitledi" için birer yüzde yazmıştık. Şimdi ara basamağı merak ediyoruz. Kapının önünde çırakların beklediği bilgisi, birkaç kelime sonra yapılacak tahmine nasıl ulaşır?

Bu soruyu **Transformer** adı verilen bir sinir ağı mimarisi üzerinden inceleyelim. Mimari derken modelin hesap yapan parçalarının nasıl düzenlendiğini ve birbirine nasıl bağlandığını kastediyoruz. İnceleyeceğimiz dil modelinde metin boyunca her token'ın bir yeri vardır. Her yer için o anki bilgiyi taşıyan bir grup sayı tutulur. Bir katman bu sayıları alır, farklı yerlerden gelen bilgileri birleştirir, sonucu işler ve sonraki katmana verir. Birkaç katman sonra "kapısını"nın bulunduğu yerdeki sayılar, öncesinde yazılanlardan da iz taşır. Model bir sonraki token'ın puanlarını hesaplarken bu son konumu kullanır.

Bunu izleyebilmek için önce tek bir konumun elinde hangi sayılarla işe başladığına bakalım. Örneğimizde "kapısını", "açtı" ve "kilitledi" kelimelerinin her birini tek token sayacağız.

17 numaralı satırda ne var?

Ustanın defterinde her metal parça için sırasıyla *uzunluk, genişlik ve kalınlık* yazdığını düşünelim. Bir parçanın ölçülerini milimetre cinsinden (40; 10; 2) diye kaydedebiliriz. Burada üç sayıyı birlikte kullanıyoruz ve her birinin sırası önemli. Genişliği okumak isteyen hep ikinci sayıya bakar. Bu bölümde **vektör** sözcüğünü böyle sıralı

bir sayı listesi anlamında kullanacağız. Listedeki her sayıya bir *bileşen* denir.

Geçen bölümde “atölyenin” kelimesini küçük sözlüğümüzde “atölye” ve “nin” diye iki parçaya ayırmıştık. Ardından “kapısını” gelince tokenizer sırasıyla 17, 42 ve 58 numaralarını veriyor. Model her numarayı bir satır adresi olarak kullanıyor. Aşağıda üç bileşenli bir örnek tablo var. Hesabı izleyebilmek için sayıları biz seçtik.

Token	Satır numarası	1. bileşen	2. bileşen	3. bileşen
atölye	17	0,6	−0,2	0,4
nin	42	−0,1	0,7	0,2
kapısını	58	0,3	0,5	−0,4

İlk numara 17. Parmağımızı o satır boyunca gezdirirsek (0,6; −0,2; 0,4) vektörünü okuruz. “Atölye” konumunda modelin hesabına bu üç sayı girer. Sıradaki 42 için başka bir üçlü okuruz. Böylece “nin” parçasının da kendine ait bir başlangıç temsili olur. Son olarak 58 numaralı satırdan “kapısını” için üçüncü vektörü alırız. **Üç token girdik, her konum için bir vektör elde ettik.** Bu tablodan seçilen başlangıç vektörüne *embedding* denir.

> **Gerçek bir modelde bir satır ne kadar uzun?** Biz 17 numaralı satırı parmağımızla izleyip üç sayı okuduk. GPT-2’nin küçük sürümünde aynı işlemi yapsaydık bir satırdan 768 sayı okurduk. Dolayısıyla 20 token’lık bir metin için model, başlangıçta her biri 768 bileşenli 20 vektör alır. GPT-2 model yapılandırması

Ustanın defterinde ikinci sütunun anlamını biliyorduk. Peki “atölye” satırındaki −0,2 neyi ölçüyor? Modeli kurarken ikinci sütuna “mekân” ya da “işçilik” gibi bir görev vermedik. Eğitim sırasında bir sonraki token için kayıp hesaplanırken geri yayılım, tablodaki sayıların ayarlanmasına da katkı sağladı. Bir token’ın dilde nasıl kullanıldığına ilişkin bilgi böylece birçok bileşene dağılabilir. Bir satırı okuduğumuzda elimizde, sonraki katmanların birlikte kullanmayı öğrendiği sayılar var.

Şimdi baştaki iki metinde “kapısını”nı bulalım. İkisinde de 58 numaralı satırı okuyoruz. O satır bize token’ın kimliğine bağlı başlangıcı veriyor. Peki aynı token’ın metnin *neresinde* geçtiğini hesaba nasıl katacağız?

Kelimelerin yeri de bilgi taşır

Atölyenin camındaki notta “Önce ölç, sonra kes” yazıyor. Aynı kelimelerle “Önce kes, sonra ölç” de yazabiliriz. İkinci talimatı izleyip parçayı kısa keserseniz iş işten geçmiş olur. Oysa iki notta kullanılan kelimelerin listesi aynı.

Model her token’ın dizideki yerini de hesaba katar. Özellikle “ölç”ün “önce” ve “sonra”ya göre nerede durduğu iki notta farklıdır. Transformer modellerinde bu bilgiyi hesaba katmanın çeşitli yolları vardır. Bizim burada izleyeceğimiz fikir şu: İki konum arasındaki ilişki hesaplanırken hem bu konumlarda hangi token’ların bulunduğu hem de birbirlerine göre nerede durdukları sonucu etkileyebilir.

Önceki cümleden bilgi nasıl geliyor?

“Kapısını” konumundayız. Bu konumun elindeki sayıları güncellerken önceki konumlardan da bilgi almak istiyoruz. Bunun basit bir yolu hepsinin ortalamasını almak olurdu. Ama her cümlede her konuma eşit pay vermek zorunda mıyız? **Dikkat** (*attention*) mekanizması, her konumdan taşınacak bilginin payını o anki bağlama göre hesaplamanın bir yoludur.

Kâğıt üzerinde küçük bir deney yapalım. “Kapısını” konumuna bilgi taşıyabilecek yerlerden ikisini seçelim. Örneği kolaylaştırmak için “çıraklar” ve “bekliyordu” birer token olsun ve deneyimizde yalnızca bu iki konuma pay verelim.

Önce taşınacak sayıları hazırlayalım. Bir vektörün *boyutu*, içinde kaç sayı bulunduğudur. Küçük modelimiz ana hesabı her konum için üç sayıyla sürdürüyor. Dikkat katmanının içinde ise iki sayılık bir ara hesap yapmayı seçelim. Bu iki genişlik model tasarlanırken belirlenir. Üç sayıdan iki sayı hesaplamanın nasıl mümkün olduğunu görmek için tek bir kural yeter.

Bu katmana gelen üç sayıya sırasıyla *x, y, z* diyelim. İlk ara sayı için *x* + *z*, ikinci ara sayı için *y* + *z* hesaplayalım. İki ayrı hesap yaptığımız için iki sonuç elde ederiz. Üçüncü sayı her iki hesaba da katılır. Gerçek bir model bu tür toplamların katsayılarını eğitimde öğrenir. Biz işlemi elle izleyebilmek için katsayıları ve aşağıdaki girdileri seçtik.

Konum	Katmana gelen üç sayı	Bu konumdan taşınabilecek iki sayı
çıraklar	(1; 3; 1)	(1 + 1; 3 + 1) = **(2; 4)**
bekliyordu	(6; 2; 2)	(6 + 2; 2 + 2) = **(8; 4)**

Elimizde hâlâ iki ayrı konum var. Her birinin aktarabileceği içeriği iki bileşenle ifade ettik. Şimdi tablonun son sütunundaki **2 ve 8** sayılarını, yani iki vektörün ilk bileşenlerini izleyelim.

Şimdi her konumdan ne kadar alacağımızı belirleyelim. İkisine eşit pay verseydik ilk bileşen için (2 + 8) / 2 = 5 elde ederdik. Bu kez "çıraklar" konumuna %75, "bekliyordu" konumuna %25 pay verelim. Ardından yalnızca payları yer değiştirip aynı hesabı tekrarlayalım.

"Çıraklar"ın payı	"Bekliyordu"nun payı	Aktarılan ilk bileşen
%75	%25	0,75 × 2 + 0,25 × 8 = **3,5**
%25	%75	0,25 × 2 + 0,75 × 8 = **6,5**

Sonucu yeniden vektör olarak okuyalım. İkinci bileşen için de aynı işlemi yaparız. İki kaynakta da bu bileşen 4 olduğu için hangi payları kullanırsak kullanalım sonuç 4 çıkar. Demek ki ilk denemede "kapısını" konumuna (3,5; 4), ikincide (6,5; 4) aktarılıyor. Bu, son konumun kendi temsilini güncellerken kullanacağı bilgidir.

Deneyde payları biz değiştirdik ve etkisini gördük. Gerçek bir cümlede hangi konumlara ne kadar pay verileceğini model nasıl hesaplıyor?

Yüzde 75'i kim belirliyor?

Model her konumdaki temsilden, eğitimde öğrenilmiş dönüşümlerle üç ayrı vektör hesaplar. Bunların adları *query*, *key* ve *value*. Üçünün görevini tek bir konumdan izleyelim. "Kapısını" konumundaki **sorgu** (*query*), bu konumun görebildiği her konumun **anahtarıyla** (*key*) karşılaştırılır. Karşılaştırmadan her konum için bir uyum puanı çıkar. Bu puanları birazdan adım adım göreceğimiz *softmax* işlemi paylara çevirir. Az önceki iki konumlu deneyde, puanlar üstel dönüşümden sonra 3 ve 1 olsaydı paylar 3/4 ile 1/4 çıkardı. Böylece %75 ve %25'i elde ederdik. Ardından bu paylarla çarptığımız (2; 4) ve (8; 4) ise **değer** (*value*) vektörleridir.

Eğitim sırasında öğrenilen şey, bu sorguları, anahtarları ve değerleri üreten dönüşümlerdir. Modeli kullanırken yeni bir cümle verdiğimizde dönüşümler aynı kalır, ama girdileri değişir. Dolayısıyla dikkat payları ve taşınan değerler yeniden hesaplanır. Bir sonraki token'ı tahmin eden modelde her konum kendisine ve solundaki konumlara erişebilir. Böylece "çıraklar" bilgisi "kapısını" konumuna ulaşabilir.

Burada bir boyut kontrolü yapalım. "Kapısını" konumunun ana temsili üç sayı, dikkatten gelen katkı ise örneğimizde (3,5; 4).

Bunları bileşen bileşen toplayabilmek için katkıyı önce üç sayı hâlinde ifade etmemiz gerekir. Dikkatin **çıkış dönüşümü** bunu sağlar. Örneğin iki girdiye u ve v deyip üç çıktı olarak u, v ve $v - u$ hesaplarsak (3,5; 4) vektöründen (3,5; 4; 0,5) elde ederiz. Artık bu katkının her bileşenini "kapısını" konumundaki karşılığına ekleyebiliriz. Burada da katsayıları kolay hesap için seçtik. Bu işlem, dikkatten toplanan bilgiyi ana temsilin kullanabileceği biçime getirir. Katmandaki sinir ağı da bu temsili dönüştürür. Bir sonraki katman artık bu işlenmiş sayılarla çalışır. Böylece önceki cümleden gelen bir ipucu, daha sonraki bir katmanda başka bir ipucuyla birlikte kullanılabilir. **"Kapısını" için girişte aynı satırı okumuştuk. Çıkışa doğru ise elimizde o metne özgü bir vektör var.**

Bir vektörden "açtı" olasılığına

Son konumun vektörünü elde ettik. Sözlüğümüzdeki hangi token bunun ardından gelmeye daha uygun? Çıktı katmanı her aday için, son vektörün bileşenlerini öğrenilmiş katsayılarla çarpıp toplayarak bir puan hesaplar. Bu ham puana *logit* denir. Son vektör değişirse aynı katsayılar farklı puanlar üretebilir.

Ana temsilimizi burada da üç bileşenli tutalım. Çıktı hesabını tek başına incelemek için bütün katmanlar tamamlandığında A metninin son vektörünün (1; 0; 1), B metnindekinin (0; 1; 1) olduğunu varsayalım. Bunlar son katman için seçtiğimiz örnek değerlerdir.

İki aday token'a puan veren küçük bir çıktı katmanı düşünelim. "Açtı" için birinci ve üçüncü bileşenleri, "kilitledi" için ikinci ve üçüncü bileşenleri toplasın. Aynı kuralları iki metne de uygulayalım.

Metin	Son konumun vektörü	"açtı" puanı	"kilitledi" puanı
A	(1; 0; 1)	1 + 1 = **2**	0 + 1 = **1**
B	(0; 1; 1)	0 + 1 = **1**	1 + 1 = **2**

Burada üç sayıdan iki puan çıkmasının sebebi, iki adayı puanlamamız. Aynı üç bileşenli temsil ile yüz adayı değerlendirseydik, her adayın kendi katsayılarını kullanarak yüz puan hesaplardık.

Softmax bu puanlarla ne yapıyor?

A metninde puanlarımız 2 ve 1. Bunları doğrudan toplama bölmek ilk bakışta yeterli görünebilir. Ama modelin puanları negatif de olabilir. Örneğin 1 ve −1 çıkarsa toplam sıfır olur. Her türlü puanı

pozitif değerlere çeviren ve büyük puanın önde kalmasını sağlayan bir işleme ihtiyacımız var.

Softmax önce her puan için e sayısının o puana karşılık gelen kuvvetini alır. İkinci bölümdeki doğal logaritmadan hatırlayacağımız gibi e yaklaşık 2,718'dir. Üs negatif olduğunda da sonuç pozitiftir. Örneğin e^{-1} yaklaşık 0,37 eder. Ardından bu pozitif sayıların toplamını alır, her birini toplama böleriz.

Aday	Ham puan	Üstel dönüşüm	Toplam içindeki pay
açtı	2	$e^2 \approx 7{,}39$	7,39 / 10,11 ≈ **%73**
kilitledi	1	$e^1 \approx 2{,}72$	2,72 / 10,11 ≈ **%27**

10,11'in nereden geldiğini parmağınızla izleyebilirsiniz. İki satırın üçüncü sütununu topladık. Son sütunda da aynı toplamı adaylar arasında paylaştırdık. Böylece büyük puan büyük payı aldı ve payların toplamı %100 oldu. Gerçek modelde bu toplama sözlükteki bütün adaylar girer.

Küçük bir deneme yapalım. İki puan da 1 olsaydı iki adayın payı da %50 olurdu. Şimdi yalnızca "açtı"nın puanını 2'ye çıkarın. Onun üstel değeri e katına çıkar. "Kilitledi"nin ham puanı hâlâ 1 olduğu hâlde payı %50'den %27'ye iner. **Bir adayın olasılığı, diğer adaylarla yarıştığı için değişir.** B metninde ise iki puan yer değiştirdiğinden bu kez %73'lük pay "kilitledi"ye gider.

Şimdi iki yüzdeyi yan yana düşünelim. Dikkat hesabındaki **%75**, bir giriş konumundan alınacak değeri hangi katsayıyla çarpacağımızı söylüyordu. Çıkıştaki **%73** ise sıradaki token için "açtı"ya verilen olasılık. İlkinde payları **metindeki konumlar** arasında, ikincisinde **sözlükteki olası devamlar** arasında dağıttık.

Başlangıçtaki okuma deneyine dönersek, iki hikâyenin aynı kelimelerini görüp farklı devamlar beklememizin modelde bir karşılığı var. İlk cümleyi değiştirince "kapısını" hâlâ aynı sözlük satırından başlıyor. Ama o konuma taşınabilecek bilgiler değişiyor. Çıktı katmanına farklı sayılar ulaştığında, "açtı" ile "kilitledi"nin puanları da değişebiliyor. Önceki bölümde tabloya kendimiz yazdığımız yüzdelerin arkasında artık takip edebildiğimiz bir hesap var.

Bölüm 4 · Sohbet modellerinde roller, bağlam ve cevap üretimi

Sıradaki token'ı tahmin eden model sorumuza nasıl cevap veriyor?

> Usta, "Öğleden sonra müşterinin fabrikasında çalışacağım. Ben yokken atölye açık kalmasın," dedi. Önlüğünü astı, atölyenin kapısını kilitledi ve yola çıktı.

Geçen bölümde hikâyeyi "kapısını" sözcüğünde durdurmuş, modelin bir sonraki sözcük olarak "kilitledi"ye nasıl yüksek bir puan verebileceğini incelemiştik. Bu kez hikâyeyi sonuna kadar verelim ve altına bir soru yazalım.

> Usta kapıyı neden kilitledi?

"Kilitledi" zaten metinde yazıyor. Biz şimdi modelden iki bilgiyi bir araya getirmesini bekliyoruz. Usta öğleden sonrayı başka bir yerde geçirecek ve yokluğunda atölyenin açık kalmasını istemiyor. Peki tek seferde yalnızca bir token seçen bir sistem böyle bir cevabı nasıl kurabilir?

Cevap da bir devam biçimidir

Bir kâğıda yalnızca "Usta kapıyı neden kilitledi?" yazıp birine uzattığınızı düşünün. Karşınızdaki büyük olasılıkla soruyu cevaplar. Aynı satırı bir Türkçe çalışma kitabında, "Aşağıdaki soruları İngilizceye çeviriniz" yönergesinin altında görseydi "Why did the master lock the door?" diye yazabilirdi. Soru aynı. Kâğıdın üstündeki birkaç sözcük, ondan beklenen işi değiştirdi.

Dil modelinin ön eğitimde karşılaştığı metinlerde de bir sorunun ardından farklı şeyler gelebilir. Bir söyleşide cevap gelir. Bir sınav kâğıdında ikinci soru gelir. Bir romanda "diye sordu çırak" gelebilir. Model bunların hepsinden öğrenebilir. Dolayısıyla bir soruya cevap yazmak, sıradaki token'ı tahmin ederek yapılabilecek işlerden biridir. Sohbet uygulamasında ise daha belirgin bir beklentimiz var. Soruyu biz sorduğumuzda modelin *asistan olarak* cevap vermesini istiyoruz.

Model asistanın sırasını nasıl öğreniyor?

Yaygın bir yöntemde, ön eğitimden geçmiş modele kullanıcı istekleriyle bunlara verilmiş örnek cevaplar gösterilir. Buna **talimatla ince ayar** (*instruction tuning*) denir. Eğitim örneklerinden birini şöyle düşünelim.

> **Kullanıcı:** “Çırak, elini sıcak metalden korumak için eldiven taktı.” Çırak neden eldiven taktı?
> **Asistan:** Elini sıcak metalden korumak için.

İkinci bölümdeki “Tahmin neyle karşılaştırılıyor?” sorusuna geri dönelim. Örnek cevabın ilk token’ının “Elini” olduğunu varsayalım. Model, kullanıcının mesajını ve asistanın sırasının geldiğini gösteren işareti görür. Sıradaki token’lar için olasılık hesaplar. Eğitim sırasında karşılaştırma yapacağımız hedef bellidir, çünkü örnek cevabı elimizde tutuyoruz. Sonraki konumda modele doğru başlangıç olan “Elini” de verilir ve bu kez onun ardından gelen token hedef alınır. Böylece örnek cevabın her parçası bir eğitim hedefi sağlar.

İkinci bölümden tanıdığımız kayıp hesabı burada da işe yarar. Örnek cevabın token’larına düşük olasılık verilmişse kayıp büyür. Eğitim adımında parametreler bu kaybı azaltacak yönde ayarlanır. Çok sayıda farklı örnekte kimi kullanıcı bir neden sorar, kimi çeviri ister, kimi de bir metni kısaltır. Bu çeşitlilik sayesinde model, yeni bir soruyla karşılaştığında da isteğin biçimine uygun cevaplar üretebilir. Bizim ustanın hikâyesinin eğitim örnekleri arasında bulunması gerekmez.

Ekranda görünen soru modelin girdisinin bir parçası

Atölye örneğine dönelim. Siz sohbet kutusuna hikâye ile soruyu yazıp “Gönder”e bastığınızda uygulama, modele kimin ne söylediğini de bildirir. Aşağıda, modele hazırlanabilecek girdiyi okunabilir bir biçimde gösteriyoruz. Gerçekte rolleri ve mesaj sınırlarını belirtmek için kullanılan işaretler modele göre değişir.

Rol	Mesaj
Sistem	Okurun sorusunu verilen hikâyeye dayanarak tek cümlede cevapla.

Rol	Mesaj
Kullanıcı	Usta, “Öğleden sonra müşterinin fabrikasında çalışacağım. Ben yokken atölye açık kalmasın,” dedi. Önlüğünü astı, atölyenin kapısını kilitledi ve yola çıktı. Usta kapıyı neden kilitledi?
Asistan	*Üretim burada başlar.*

“Sistem” satırındaki yönerge cevap için iki beklenti oluşturuyor. Bilginin kaynağı hikâye olacak ve cevap tek cümle sürecek. Kullanıcı satırı hem üzerinde çalışılacak metni hem de soruyu içeriyor. En sondaki rol işareti de üretilecek metnin kimin sözü olarak okunacağını belirtiyor. Modelin o anda kullanabildiği bu bilgilere **bağlam** diyoruz.

Burada küçük bir deney yapabiliriz. Hikâyenin tek sözcüğünü değiştirmeden son soruyu “Usta öğleden sonra nerede çalışacak?” yapalım. “Müşterinin fabrikasında” artık kendi başına yeterli bir cevap. İlk soruda ise aynı yer bilgisi bir gerekçenin parçasıydı. Üçüncü bölümde gördüğümüz dikkat mekanizması burada da devrededir. Sorudaki “nerede” ile “neden” farklı girdiler oluşturur. Bu fark, önceki konumlardan taşınan bilgileri ve cevap başlarken aday token’lara verilen puanları değiştirebilir.

Bir token’dan bir cümleye

İlk soruya modelin şöyle bir cevap ürettiğini düşünelim.

> Usta, müşterinin fabrikasına gideceği için atölyenin açık kalmasını istemedi.

Bu cümleyi biz bir bütün olarak okuyoruz. Model açısından birkaç ayrı anda durup bakalım. Tabloda okunabilirlik için birden fazla token’ı kapsayabilen metin parçaları gösteriyoruz.

Durduğumuz an	Bir sonraki token hesaplanırken elde ne var?
Cevap henüz başlamadı	Yönerge, hikâye, soru ve asistanın sırasını belirten işaret. “Usta” da “Atölyenin” de birer olası başlangıç.

Durduğumuz an	Bir sonraki token hesaplanırken elde ne var?
“Usta, müşterinin” yazıldı	Başlangıçtaki bağlama şimdi bu iki sözcük ve virgül de eklendi. Hikâyedeki “müşterinin fabrikasında” ifadesi, “fabrika” ile başlayan bir devamı destekliyor.
“Usta, müşterinin fabrikasına gideceği için” yazıldı	“İçin” ile biten parça, bir neden bildirdi. Cümleyi tamamlayacak devam için ustanın atölye hakkındaki sözü hâlâ bağlamda.

Her adımda model olasılıkları yeniden hesaplar ve üretim yöntemi bu olasılıklardan bir token seçer. Seçilen token bir sonraki adımın girdisine katılır. Örneğin ilk sözcük “Atölyenin” olsaydı, “açık kalmasını istemediği için kapıyı kilitledi” gibi farklı bir cümle kurulabilirdi. Böylece aynı gerekçe birkaç doğru biçimde ifade edilebilir. Yanıt sonunu bildiren özel token seçildiğinde uygulama üretimi bitirebilir.

Eğitim ile kullanım arasındaki küçük ama önemli fark tam burada görülür. İnce ayar sırasında doğru devamı örnek cevaptan alabiliyorduk. Siz soru sorduğunuzda uygulamanın elinde karşılaştıracağı bir cevap anahtarı yoktur. Model bir önceki adımda kendi ürettiği parçadan devam eder. Hikâyede ustanın gideceği yeri değiştirirseniz cevap da değişebilir. Bunun için modelin parametrelerinin yeniden ayarlanması gerekmez. **Değişen şey, o cevap hesaplanırken modelin önünde bulunan bilgidir.**

Hikâyeden bir cümleyi çıkarırsak

Şimdi ustanın kendi sözlerini metinden silelim. Geriye şu kalsın.

> Usta önlüğünü astı, atölyenin kapısını kilitledi ve yola çıktı.
> Usta kapıyı neden kilitledi?

Siz bu kez ne cevap verirdiniz? “Atölyeden ayrılıyordu” diyebiliriz. “İçerideki aletler çalınmasın diye” de akla yatkın bir tahmin. Fakat hangimizin aklına “müşterinin fabrikasına gidecekti” gelirdi? Bu ayrıntıyı biz önceki hikâyeyi okuduğumuz için hatırlıyoruz. Deneyi bir sohbet uygulamasında yaparsanız kısa metni **yeni bir sohbette, önceki hikâyeyi vermeden** sorun. Aynı konuşmada sorarsanız, uygulama ilk hikâyeyi de bağlama ekleyebilir.

Kısa metne dayanarak verilebilecek ölçülü bir cevap şu olurdu. “Usta atölyeden ayrılırken kapıyı kilitledi. Bunu hangi nedenle yaptığı hikâyede belirtilmiyor.” Bir model yine de “Aletleri hırsızlardan korumak için” diye cevap verebilir. Bu açıklamayı gündelik deneyimimize uydurmak kolaydır. Ama hikâyeye dönüp “Hangi sözden bunu çıkardık?” diye sorunca gösterebileceğimiz bir dayanak bulamayız. Modelin bir sonraki token’a verdiği yüksek olasılık, o token’ın bağlama uygun bir devam olabileceğini ifade eder. Anlatılan olayın gerçekten öyle yaşandığını doğrulamaz.

Bir de şu soruyu soralım. **Atölyenin kapısı şu anda kilitli mi?** İlk hikâyeyi eksiksiz geri getirsek bile yalnızca ustanın çıkarken ne yaptığını biliyoruz. Belki çırak beş dakika sonra gelip kapıyı açtı. Bugünkü durumu öğrenmek için kapıyı gören birine sormamız ya da örneğin akıllı kilidin son durumunu okumamız gerekir. Peki bir sohbet uygulaması böyle bir dış kaynağa nasıl ulaşır ve oradan gelen bilgi modelin cevabına nasıl girer? Bir sonraki bölümde atölyenin kapısını bu kez dış dünyaya açacağız.

Bölüm 5 · Araç çağrıları ve verinin güncelliği

Atölyenin kapısı şu anda kilitli mi?

Saat 15.20. Usta müşterinin fabrikasında çalışıyor. Öğle arasında atölyeden çıkarken kapıyı kilitlediğini biliyoruz. Şimdi çırağın telefonundan bir mesaj geliyor.

> Atölyenin kapısı şu anda kilitli mi?

Bir an için telefondaki asistanın yerinde olduğunuzu düşünün. Önünüzde geçen bölümdeki hikâye var. Cevap vermek için hikâyeyi istediğiniz kadar okuyabilirsiniz. Peki ya siz okurken ustanın ortağı kendi anahtarıyla kapıyı açtıysa? Ortağın geldiği bir öğleden sonra ile hiç kimsenin gelmediği bir öğleden sonra, önünüzdeki kâğıtta kelimesi kelimesine aynı görünüyor. Bu iki olasılığı ayırmak için dışarıdan bir gözlem gerekiyor.

Önce en basit yolu deneyelim. Çırak kapıya kadar yürüsün, kilidi kontrol etsin ve sohbete “Saat 15.20’de kontrol ettim, kapı kilitli” yazsın. Model artık “Çırağın 15.20’deki kontrolüne göre kapı kilitli” diye cevap verebilir. Geçen bölümden tanıdığımız bir şey oldu. Konuşmaya yeni bir mesaj eklendi ve bu mesaj cevaba dayanak sağladı. Şimdi çırağı oraya kadar yürütmeden aynı bilgiyi nasıl elde edebileceğimize bakalım.

Model hangi bilgiyi isteyebileceğini nereden biliyor?

Atölyeye bir akıllı kilit takıldığını düşünelim. Kilidin üzerindeki algılayıcı, sürgünün konumunu okuyabiliyor. Sohbet uygulamasını yazan kişi de bu okumayı isteyen küçük bir işlev hazırlamış. İşlev, hangi kapının sorulduğunu alıyor ve o kapıdaki cihazın bildirdiği durumla ölçüm saatini döndürüyor. Uygulamanın modelin kullanımına sunduğu böyle bir işleve **araç** (*tool*) diyoruz.

Birinin size “Bu numarayı arayıp atölyenin ana kapısının durumunu sorabilirsin” dediğini düşünün. Artık bilgiye hangi yoldan ulaşacağınızı biliyorsunuz. Kapının durumunu öğrenmeniz içinse telefonun öbür ucundan bir cevap gelmesi gerekiyor. Modelin önüne konan araç tanımı da kabaca böyle okunabilir.

Aracın adı	Ne işe yarar?	Hangi bilgiyi ister?
`kilit_durumunu_oku`	Belirtilen kapının kilit durumunu ve ölçüm saatini getirir.	Kapının adı

Uygulama, modele çırağın sorusuyla birlikte bu tanımı da sunar. Örneğimizde atölyenin tek giriş kapısı var ve uygulamada “atölye ana kapısı” adıyla kayıtlı. Model böylece hem hangi işlemin işine yarayacağını hem de hangi kapıyı belirtmesi gerektiğini girdisinde görebilir.

“Kilide bir soralım” nasıl bir çıktıdır?

Geçen bölümde modelin asistan sırası geldiğinde bir cevap üretmeye başladığını görmüştük. Araç kullanabilen bir model, o sırada uygulamaya yönelik bir istek de üretebilir. Buna **araç çağrısı** (*tool call*) denir. İsteği okunabilir biçimde yazarsak iki parçasını kolayca görebiliriz.

> **İstenen işlem:** kilit_durumunu_oku
> **Kapı:** atölye ana kapısı

Model bu isteği üretirken de bağlamdan yararlanır. “Şu anda” ifadesi güncel bir gözlem gerektiğine, “kapı” sözcüğü de kilit aracının işe yarayabileceğine işaret eder. Araç tanımı, isteğin nasıl yazılacağını gösterir. Burada bir an duralım. Modele tam bu noktada “Peki sonuç ne?” diye sorsak elimizde hâlâ yalnızca ustanın öğle vakti ne yaptığı var. Yukarıdaki iki satırda kapının kilitli olduğuna dair bir bilgi görebiliyor musunuz?

Uygulama bu isteği alır, adı verilen işlevi çalıştırır ve cihazdan cevap bekler. Örneğimizde cihaz sürgünün konumunu okur. İşlevin döndürdüğü sonucu şöyle gösterelim.

> **Kapı:** atölye ana kapısı
> **Kilit durumu:** kilitli
> **Ölçüm saati:** bugün 15.20

“15.20’de kilitli” bilgisi ilk kez burada ortaya çıktı. Uygulama sonucu konuşmaya bir *araç mesajı* olarak ekler ve modeli yeniden çağırır. Model bu kez çırağın sorusunu, kendi istediği işlemi ve işlemin sonucunu birlikte görür. Çırağın ekranına şu cevap gelebilir.

> Evet. Ana kapının kilidi, saat 15.20’de kapıyı kilitli olarak bildirdi.

Çırak ekranda bir soru ve bir cevap görüyor. Arada model iki kez çalıştı. İlkinde neyin sorgulanacağını belirten isteği üretti. İkincisinde, artık ölçümü de içeren bağlamdan yararlanarak cevabı yazdı. Birkaç saniye önce modelin elinde bulunmayan bir bilgi böylece cümlenin içine girebildi. Bunun için yeniden eğitim gerekmedi.

Tek bir satırı değiştirirsek

Şimdi uygulamanın saati yine 15.20 olsun. Çırağın sorusunu, hikâyeyi ve istenen işlemi aynen bırakalım. Yalnızca araçtan dönen sonucu değiştirelim. Tablonun sağ sütununu elinizle kapatıp her satırda çırağa ne yazacağınızı düşünün.

Aracın döndürdüğü sonuç	Bu sonuca dayanarak verilebilecek cevap
Kilitli. Ölçüm bugün 15.20’de yapıldı.	Son ölçüme göre ana kapı saat 15.20’de kilitli.
Kilitli. Son ölçüm bugün 12.05’te yapıldı.	Öğlen 12.05’te kilitliymiş. O zamandan beri durumun değişip değişmediğini bilmiyoruz.
Cihaza ulaşılamadı. Kayıtlı ölçüm yok.	Kilide ulaşamadığım için kapının şu anki durumunu doğrulayamıyorum.

İkinci satırda cevap telefona **15.20’de** geldi. İnsan tam da bu yüzden kilidin az önce kontrol edildiğini düşünebilir. Oysa cihazın son bildirdiği durum **12.05’e** ait. Çırak “şu anda” diye sorduğu için bu üç saat on beş dakika cevabın anlamını değiştiriyor.

Üçüncü satırda ise bağlantının başarısız olması, sürgünün her iki konumuyla da bağdaşır. Model “Usta çıkarken kilitlemişti, demek ki kilitli” diye yazsaydı, ölçüm alamadığımız yerde hikâyeden bir tahmin üretmiş olurdu. Geçen bölümde sorduğumuz “Bu cevabı hangi cümleye dayandırdık?” sorusunu şimdi biraz genişletebiliriz. **Hangi cihaz, tam olarak neyi, ne zaman ölçtü?**

Son olarak çırağın aynı akşam bir soru daha sorduğunu düşünelim. “Kapı hâlâ kilitli mi?” Öğleden sonraki cevap sohbet ekranında duruyor. Hatta uygulama o cevabı modelin önüne yeniden koymuş da olabilir. Saat 15.20’deki ölçüm akşama kadar eskidiğine göre, bu soruyu cevaplarken modelin bir kez daha istemesi gereken şey nedir?

Bölüm 6 · Belgelerde arama ve kaynağa dayalı cevap üretimi (RAG)

Model, bakım defterinde cevabı nasıl buluyor?

Kasım başında çırak, atölyenin girişinde eski kapının pirinç tokmağını buluyor. Usta onu söküp bir kenara ayırmış. Çırak telefondaki asistana soruyor.

Geçen ay ana kapıyı neden değiştirdik?

Geçen bölümde kullandığımız akıllı kilit şu anda sürgünün konumunu okuyabiliyor. Ekim ayında ustanın niçin yeni bir kapı taktırdığını ise bakım defterine sormak daha yerinde olur. Gelin, önce defteri biz açalım. Bir ayrıntıyı bilmemiz işimizi kolaylaştıracak. Atölyenin ana kapısından malzemeler teslim alındığı için bakım ekibi buraya “mal kabul girişi” de diyor.

Tarih	Bakım defterindeki kayıt
3 Ekim	Ana kapının kilidi yağlandı. Anahtar birkaç kez çevrilerek kontrol edildi.
14 Ekim	Mal kabul girişindeki kanat eğilmişti. Düzgün kapanmadığı için söküldü ve yerine yenisi takıldı.
22 Ekim	Depo kapısının altından su girdiği görüldü. Yıpranan conta değiştirildi.

Çırağa hangi satırı gösterirdiniz? İlkinde “ana kapı” aynen geçiyor. Üçüncüsünde hem bir kapıdan hem de bir değişimden söz ediliyor. Ama çırağın “neden?” sorusuna cevap verirken parmağımızı 14 Ekim kaydındaki “düzgün kapanmadığı için” sözlerine koyabiliyoruz. Üstelik aynı kayıtta hem arızayı hem de ustanın ne yaptığını görebiliyoruz.

Defterde otuz bin kayıt olsaydı

Bu üç satırı soruyla birlikte modele verirsek geçen bölümlerden tanıdığımız bir durum oluşur. Model, kendisine sunulan metne dayanarak cevabını yazabilir. Şimdi aynı atölyenin on yıllık arşivinde otuz bin kayıt bulunduğunu düşünün. Her soru için bütün arşivi göndermek zaman ve işlem gücü gerektirir. Ayrıca modelin bir defada işleyebildiği metnin bir sınırı vardır. Peki defterin hangi sayfasını açacağız?

Önce bir bilgisayarın "ana kapı" ifadesini harfi harfine aradığını düşünelim. İlk satırı kolayca bulur. Oysa bizim cevabımızı taşıyan satırda ne "ana" ne de "kapı" yazıyor. Biz okurken iki bağlantı kurduk. Bu atölyede mal kabul girişinin neresi olduğunu biliyoruz. Ayrıca "söküldü ve yerine yenisi takıldı" sözlerinin bir değişimi anlattığını anlıyoruz. Arama sisteminin de farklı sözcüklerle anlatılmış bir işi bulabilmesini isteriz.

Bunun yollarından biri, üçüncü bölümde tanıdığımız **vektörlerden** yararlanmak. Orada bir token için bir sayı listesiyle başlamıştık. Burada aradığımız şey bütün bir bakım kaydı. Bu yüzden her kısa kaydı bir bütün olarak sayılarla temsil eden bir model kullanabiliriz. Buna *embedding modeli* denir. Arama için eğitilirken, bir soru ile ona cevap veren metin parçasının vektörlerini birbirine yakın üretecek şekilde ayarlanabilir. Buradaki "yakınlık" hesaplanabilir bir şeydir. Örneğin iki vektörün yönlerinin ne kadar benzeştiği bir puana dönüştürülebilir.

Soru gelmeden önce ve geldikten sonra

Arşivi hazırlarken uygulama her kayıt için bu sayı listesini bir kez hesaplar. Listenin yanında kaydın tarihini ve özgün metni bulmaya yarayan kayıt numarasını saklar. Böylece 14 Ekim kaydının vektörü, çırağın o gün aklına hangi sorunun geleceğini bilmeden de hazırlanabilir.

Soru geldiğinde uygulama bu kez sorunun vektörünü hesaplar. Örneğimizde aramayı ekim ayıyla sınırlandırabilir ve atölyenin yer listesinden yararlanarak sorguya "mal kabul girişi" adını da ekleyebilir. Bu küçük ayrıntı önemlidir. Bir embedding modeli "yerine yenisi takıldı" ile "değiştirildi" arasında ilişki kurabilir. Ama bizim atölyemizde iki adın aynı kapıyı gösterdiğini nereden bilecek? Arama sırasında yer listesini kullanmak bu özel bilgiyi sisteme bizim vermemizi sağlar.

Uygulama soru vektörünü aday kayıtların vektörleriyle karşılaştırır. Yüksek puan alan birkaç kaydın **özgün metnini** bulur ve tarihleriyle birlikte sohbet modelinin bağlamına ekler. Burada puanın işini bitirdiği yere dikkat edin. Puan, defterin hangi sayfasını açmaya değer bulduğumuzu gösterir. Okura “kapı kanadı eğilmişti” diyebilmemizi sağlayan şey ise açılan sayfada yazan cümledir.

14 Ekim kaydı getirildiğinde sohbet modeli şöyle cevap verebilir.

> 14 Ekim tarihli bakım kaydına göre ana kapının kanadı eğilmiş ve kapı düzgün kapanmaz olmuş. Bu yüzden kanat sökülüp yenisi takılmış.

Çırak “kapıyı değiştirdik” demişti. Kayda bakınca daha ince bir ayrıntı da ortaya çıktı. Usta özellikle *kanadın* yenilendiğini yazmış. Cevapta tarihi görmek, çırağın dönüp o ayrıntıyı kendi gözleriyle kontrol etmesine imkân verir.

Bu yönteme **RAG** (*retrieval-augmented generation*) deniyor. İngilizce adın ilk kısmı ilgili kayıtların aranıp getirilmesini, son kısmı da bu kayıtlarla genişletilmiş bağlamdan cevabın üretilmesini anlatıyor. Vektörle arama, kayıtları getirmenin yollarından biri. Defter düzenli tutulmuşsa bazen tarih ve parça numarasıyla yapılan basit bir arama da aynı işi görür. Her iki durumda da yarın yeni bir bakım kaydı eklenip aranabilir hâle getirildiğinde, sohbet modeli o kaydı kullanabilir. Bunun için modeli yeniden eğitmek gerekmez.

Doğru sayfa masaya gelmezse

Şimdi küçük bir deney yapalım. Yukarıdaki tabloda ikinci ve üçüncü satırı elinizle kapatın. Aramanın yalnızca 3 Ekim tarihli yağlama kaydını getirdiğini düşünün. Çırağın sorusu hâlâ aynı. Önünüzdeki tek satıra bakarak nasıl cevap verirdiniz?

“Kilidi bozuk olduğu için kapıyı değiştirmişler” kulağa makul gelebilir. Bir kez daha kayda bakalım. Kilit yağlanmış. Yağlamanın ardından anahtar çevrilmiş. Kaydı yazan kişi kilidin bozuk olduğunu söylemiş mi? Peki bu işlemi iki hafta sonraki değişime bağlayan bir şey yazmış mı?

Elimizdeki kanıtla verebileceğimiz cevap şöyle olurdu. “Bana gelen 3 Ekim kaydında kilidin yağlandığı yazıyor. Kapının niçin değiştirildiği bu kayıttan anlaşılmıyor.” Şimdi elinizi kaldırın. **“Bu kayıtta gerekçe yok” diyebildik. “Arşivde gerekçe yok” deseydik yanılmış olacaktık.** Böyle bir durumda uygulamanın “mal kabul girişi” adıyla aramayı yeniden denemesi, aynı eksik metin üzerinde daha uzun bir cevap üretmekten çok daha işe yarayabilir.

Doğru kayıt bulundu. Peki ne kadarı getirildi?

Bu kez arama doğru sayfayı bulsun. Fakat defteri bilgisayara aktarırken her cümleyi ayrı bir parça olarak sakladığımızı düşünelim. Arama sistemi 14 Ekim kaydından yalnızca şu parçayı getirsin.

> Düzgün kapanmadığı için söküldü ve yerine yenisi takıldı.

Bu cümleyi tek başına bir kâğıda yazıp atölyeyi tanımayan birine verin. Neden söküldüğünü söyleyebilir. Peki *neyin* söküldüğünü ya da niçin düzgün kapanmadığını nereden bilecek? Çırağın sorusunda "ana kapı" geçtiği için eksik yeri öyle tamamlamak kolay. Ama aynı cümle bir dolabın bakım kaydından da çıkabilirdi.

Şimdi kâğıdın başına bir önceki cümleyi geri koyun. "Mal kabul girişindeki kanat eğilmişti." Birkaç sözcük ekledik ve artık hem sökülen parçayı hem de kapanma sorununun nedenini gösterebiliyoruz. Uzun belgeleri aramaya hazırlarken **nereden böldüğümüz** bu yüzden önemlidir. Bizim defterimizde her tarihli bakım kaydını bütün tutmak doğal bir seçim. Bir kaydın birkaç sayfa sürdüğü başka bir defterde ise bulunan paragrafla birlikte başlığını ve komşu paragrafı getirmek daha kullanışlı olabilir.

RAG kullanan bir asistan size şaşırtıcı bir cevap verdiğinde bu deneyi hatırlayın. Önce "Modele defterden hangi satırlar gösterildi?" diye sorun. Sonra o satırları okuyup "Ben de buradan aynı cevabı çıkarabilir miydim?" diye bakın. İlk soru aramanın ne getirdiğini, ikincisi modelin getirilen bilgiyi nasıl kullandığını görmenizi sağlar.

Son olarak defteri kapatalım. Çırak bu kez telefonunda 14 Ekim sabahı çekilmiş bir fotoğraf buluyor. Fotoğrafta eski kanadın üst kenarı kasaya dayanmış, alt köşesinde ise ışık sızan bir aralık görünüyor. Yazılı kayıtta usta bize ne olduğunu kendi sözcükleriyle anlatmıştı. Şimdi elimizde yalnızca pikseller var. Bir model bu görüntüden kapının kasaya nasıl oturduğunu nasıl çıkarabilir? Ve yalnızca o sabahın fotoğrafına bakarak öğleden sonra kanadın değiştirildiğini söyleyebilir mi?

Bölüm 7 · Görüntü algılama, pikseller ve görsel özellikler

Bir model kapının altından sızan ışığı nasıl fark eder?

Çırak, 14 Ekim sabahı çektiği fotoğrafı telefonda açıyor. Görüntüyü iki parmağıyla öyle büyütüyor ki ekranda yalnızca açık renkli bir kare kalıyor. “Usta,” diyor, “kapının neresine bakıyorum?”

Usta omuz silkiyor. O kare pirinç tokmağın üzerindeki bir parıltıya da ait olabilir, kapının altından sızan gün ışığına da. Çırak görüntüyü biraz küçültünce karenin yanında başka açık renkli kareler beliriyor. Biraz daha küçültüyor. Şimdi ince bir ışık şeridinin kapının alt kenarını izlediğini, sağ köşeye doğru genişlediğini görüyoruz. Baştaki kare yerli yerinde duruyor. Ona getirdiğimiz yorumu değiştiren şey çevresi.

Çırak aynı fotoğrafı bir sohbet uygulamasına yükleyip “Kapının neresinden ışık sızıyor?” diye sorsa, model bu ilişkiden nasıl yararlanabilir?

Dört pikseli kâğıda alalım

Bir fotoğrafta her pikselin bir konumu ve o konumdaki rengi bildiren sayılar vardır. Kırmızı, yeşil ve mavi bileşenleri ayrı ayrı tutan renkli bir görüntüde bir piksel için üç sayı kullanabiliriz. Hesabı elimizle izleyebilmek için fotoğrafın gri tonlu bir kopyasını alalım. Her piksele 0 ile 255 arasında tek bir sayı verelim. 0 siyah, 255 beyaz. Sayı büyüdükçe o nokta daha açık görünür.

Kapının alt kenarına yakın, iki piksel genişliğinde ve iki piksel yüksekliğinde bir parça seçelim. Örnek değerlerimiz şöyle olsun.

	Sol piksel	Sağ piksel
Üst sıra	20	20
Alt sıra	220	220

Üst sırada iki koyu, alt sırada iki açık piksel var. Şimdi şu soruya bir sayı ile cevap verelim. *Parçanın alt yarısı üst yarısından ne kadar daha açık?* Alt sıranın ortalaması 220, üst sıranınki 20. Alttan üstü çıkarırsak **220 − 20 = 200** buluruz.

Kâğıdı çeyrek tur döndürdüğünüzü düşünün. Açık piksellerin ikisi de artık aynı sütunda. Her sırada bir açık, bir koyu piksel var. İki sıranın ortalaması da 120 olduğundan aynı hesap bu kez **120 − 120 = 0** veriyor. Oysa dört pikselin toplam parlaklığı iki durumda da aynı. Yalnızca yerleşimleri değişti.

Demek ki dört sayıyı nasıl birleştirdiğimiz, hangi farkı yakaladığımızı belirliyor. İlk hesap yatay bir aydınlık geçişine duyarlıydı. Sütunları karşılaştıran ikinci bir hesap da dikey geçişe duyarlı olurdu. Bu iki sonucu yan yana yazarsak, üçüncü bölümden tanıdığımız türden küçük bir **vektör** elde ederiz. Her iki sayı da aynı dört pikselden geldiği hâlde parça hakkında farklı bilgi taşır.

Bu hesabı modele kim öğretiyor?

Bir önceki hesabı biraz yavaşlatalım. Üst sıranın ortalamasını alırken iki pikseli topladık ve ikiye böldük. Aynı ortalamayı her pikselin yarısını alıp bu iki yarıyı toplayarak da bulabiliriz. Alt sırada da her pikselin payı yarım. Peki eksi işareti nereden geliyor? Son işlemimiz *alt sıranın ortalamasından üst sıranın ortalamasını çıkarmak*. Öyleyse üstteki iki yarımın katkısını çıkaracak, alttaki iki yarımın katkısını ekleyeceğiz.

Piksel	Parlaklık	Hesaptaki payı	Sonuca katkısı
Sol üst	20	−½	−10
Sağ üst	20	−½	−10
Sol alt	220	+½	+110
Sağ alt	220	+½	+110

Son sütunu toplarsak −10 − 10 + 110 + 110 = **200** buluruz. Tablodaki *hesaptaki payı* sütunu, birinci bölümde gördüğümüz **ağırlıklar**. Eksi işareti pikselin karanlık olduğunu söylemiyor. O konumdaki parlaklığın sonuçtan çıkarılacağını söylüyor. Örneğin dört pikselin hepsi 220 olsaydı üstten gelen −220 ile alttan gelen +220 birbirini götürürdü. Parça çok aydınlık olduğu hâlde üstten alta geçiş sıfır olurdu.

Bu örnekte ağırlıkları biz belirledik, çünkü hangi sorunun cevabını hesaplamak istediğimizi biliyorduk. Bir görüntü ağında bu ağırlıklar öğrenilebilir. Örneğin elimizde alt kenarında açıklık bulunan ve bulunmayan çok sayıda etiketli kapı fotoğrafı olsun. Ağ her fotoğraf için bir tahmin üretir. Kayıp hesabı ve geri yayılım, tahmin hatasına göre görüntüyü işleyen katmanlardaki ağırlıkların da ayarlanmasını

sağlar. Eğitimin hedefi etiketleri doğru tahmin etmektir. Sonunda hangi ağırlıkların işe yarayacağı örneklere ve göreve bağlıdır.

Bizim 200 sayımızın anlamı sınırlı. Duvara çizilmiş beyaz bir çizginin kenarında da aynı sonucu bulabiliriz. Kapının altındaki aralıktan söz edebilmek için fotoğrafın geri kalanına bakalım.

Parçalar birbirinden nasıl haberdar olur?

Vision Transformer (ViT), üçüncü bölümde metin üzerinde incelediğimiz dikkat mekanizmasını görüntü parçaları üzerinde kullanan bir mimaridir. Fotoğrafın üstüne kareli bir kâğıt koyduğumuzu düşünelim. Her kare bir görüntü parçası, İngilizce adıyla bir *patch* olsun. Hesabı küçük tutmak için karelerimiz yine ikişer piksel genişliğinde ve yüksekliğinde olsun. Model her kareye aynı sırayla bakar. Sol üst, sağ üst, sol alt, sağ alt. Bizim parçamız böylece **(20, 20, 220, 220)** listesiyle temsil edilebilir.

Parçanın içinde ne var?

Her parça için iki sayı hesaplayalım. İlki biraz önce bulduğumuz *alt ortalama eksi üst ortalama*, ikincisi *sağ ortalama eksi sol ortalama* olsun. İlk parçamızın sağında da solunda da ortalama 120 olduğuna göre ikinci sayı sıfırdır. Böylece dört parlaklık değerinden **(200, 0)** vektörünü elde ederiz. Aynı iki hesabı başka parçalara da uygulayabiliriz.

Parçadaki düzen	Dört pikselin değeri	Elde ettiğimiz vektör
Alt sıra açık	(20, 20, 220, 220)	(200, 0)
Sağ sütun açık	(20, 220, 20, 220)	(0, 200)
Her yer koyu	(20, 20, 20, 20)	(0, 0)

Gerçek bir ViT’de renkli bir parçanın bütün renk bileşenleri hesaba girer. Öğrenilen farklı ağırlıklı toplamlar, parçayı temsil edecek vektörün bileşenlerini üretir. Burada iki bileşene kolayca ad verebildik, çünkü hesapları kendimiz seçtik. Gerçek modelin öğrendiği bir bileşenin neyi yakaladığını böyle tek bir ifadeyle açıklamak güç olabilir.

Parça nerede?

Şimdi aynı koyudan açığa geçişin bir kez tokmağın yakınında, bir kez de eşikte görüldüğünü düşünün. Dört parlaklık değeri aynıysa

yukarıdaki işlem iki parça için de **(200, 0)** verir. Oysa çırağın sorusunda “nerede?” sözcüğü var. Bu bilgi nereden gelecek?

Özgün ViT tasarımında her parça yerine, parça vektörüyle aynı uzunlukta ayrı bir **konum vektörü** atanır. Bu vektörlerin değerleri eğitim sırasında öğrenilir. Toplama işlemini görebilmek için iki yere örnek sayıları biz verelim.

Parçanın yeri	İçerikten gelen vektör	O yere ait konum vektörü	Bileşenleri topladıktan sonra
Tokmağın yakınındaki kare	(200, 0)	(1, 3)	(201, 3)
Eşikteki kare	(200, 0)	(4, 2)	(204, 2)

İlk satırda 200 ile 1’i, 0 ile 3’ü topladık. **(1, 3)** burada “birinci satır, üçüncü sütun” anlamına gelmiyor. Kareyi fotoğraf üzerindeki yeriyle eşleştirdikten sonra o yere ait sayı listesini kullandık. Ertesi gün aynı yerde koyu bir parça görürsek bu kez içerikten (0, 0) gelir ve aynı konum vektörünü ona ekleriz. Böylece hem içeriği değiştirmek hem de aynı içeriği başka bir yere taşımak, sonraki katmana ulaşan sayıları değiştirebilir.

Bu aşamada eşikteki kare hâlâ yalnızca kendi piksellerinden ve kendi yerinden bilgi aldı. **Dikkat** katmanında ise o karenin vektörü diğer karelerin vektörleriyle karşılaştırılır. Üçüncü bölümde bir token için yaptığımız gibi, farklı konumlardan taşınacak bilgiye farklı ağırlıklar verilir. Örneğin komşu karelerdeki açık renkli geçişler yan yana uzanıyorsa bunlardan gelen bilgi, tek bir parıltıyla uzun bir şeridi ayırmaya yardım edebilir. Sonraki katmana geçen eşik vektörü artık komşuları hakkında da iz taşır. Birkaç katman boyunca bu işlem tekrarlanınca küçük parçalar üzerinden kapıyla kasa arasındaki ilişkiyi değerlendirmek mümkün olur.

Peki “sağ alt köşeden” sözünü kim yazıyor?

Bir görüntü modelinin çıktısı, eğitimde öğrenmesini istediğimiz işe bağlıdır. Kapı fotoğraflarıyla eğittiğimiz basit sınıflandırıcı, “açıklık var” ve “açıklık yok” seçeneklerine birer puan verebilir. Çırağın sorusu biraz daha fazlasını istiyor. Açıklığın *nerede* olduğunu Türkçe bir cümleyle anlatmamız gerekiyor.

Fotoğrafla soruyu birlikte işleyen bir sohbet modeli düşünelim. İçinde bir görüntü kodlayıcısı, iki bölüm arasında öğrenilebilir bir bağlantı ve bir dil modeli olsun. Bu, görüntü ile metni bir araya getirmenin yollarından biridir. Çırağın sorusunu bu düzenekte adım adım izleyebiliriz.

Önce fotoğraf işlenir. Görüntü kodlayıcısı az önce izlediğimiz işlemler sonucunda bir dizi vektör üretir. Bunların bazıları kapının alt kısmındaki renk geçişinden ve bu geçişin komşu karelerde nasıl sürdüğünden bilgi taşır. Bu aşamada Türkçe bir cümle kurulması gerekmez.

Sonra görsel bilgi dil modeline aktarılır. İki sayılık örnek vektörümüzü hatırlayın. Diyelim ki dil tarafı her konumda üç sayı bekliyor. Araya, iki girdiden üç ayrı ağırlıklı toplam hesaplayan küçük bir katman koyabiliriz. Böylece her görsel parça için dil tarafının işleyebileceği uzunlukta bir liste elde ederiz. Boyutların uyuşması ilk adım. Bağlantının işe yaraması için o üç sayının *hangi görsel ayrıntıları nasıl taşıdığının* da eğitimle ayarlanması gerekir.

Bunu küçük bir eğitim örneğinde düşünelim. Fotoğrafın altında “Işık nereden geliyor?” sorusu ve örnek cevap olarak “Kapının sağ alt köşesinden” cümlesi bulunsun. Dil modeli cevabın token’larına düşük olasılık verdiğinde ikinci bölümde tanıdığımız kayıp hesaplanabilir. Bu hata bağlantı katmanına kadar geri taşınarak onun ağırlıklarının ayarlanmasına yardım eder. Işığın farklı yerlerde göründüğü pek çok örnekte, görsel konumu koruyacak bir dönüşüm doğru cevapları üretmeyi kolaylaştırır. İki bölüm arasında ortak bir kullanımın nasıl öğrenilebildiğini böyle düşünebiliriz.

Çırak kendi fotoğrafını yüklediğinde dil modeli artık hem sorunun token’larından gelen vektörleri hem de fotoğraftan aktarılan vektörleri kullanabilir. “Nereden?” sorusuna cevap üretirken dikkatin bir kısmı konum bilgisi taşıyan görsel parçalara yönelirse, bu bilgi sıradaki token’ın puanına katkıda bulunur. Model “Işık kapının altından sızıyor, aralık sağ köşede daha geniş görünüyor” cümlesini token token yazabilir. İki bölümün “iletişimi” dediğimiz şeyin bu örnekte somut bir karşılığı var. Aralarından geçen, fotoğraftan hesaplanmış sayı listeleridir.

Fotoğrafın tarihini ileri alabilir miyiz?

Son bir deney yapalım. Fotoğrafın tamamını gören iki kişiden birine “Işık şeridi nerede genişliyor?”, diğerine “Usta öğleden sonra kapı kanadını değiştirdi mi?” diye sorun. Birincisi cevabını parmağıyla

fotoğrafta gösterebilir. İkincisi aynı kareyi ne kadar büyütürse büyütsün öğleden sonra yaşananları göremez.

Geçen bölümde bakım defterini okuduğumuz için biz kanadın yenilendiğini biliyoruz. Şimdi hayalimizde o kaydı fotoğrafın yanına koyalım. **“Nerede aralık vardı?”** sorusunun cevabını fotoğrafta, **“Sonra ne yapıldı?”** sorusunun cevabını tarihli kayıtta gösterebiliyoruz. İkisini aynı sohbette kullanan bir modelin cevabını değerlendirirken bizim de elimizde aynı basit yöntem var. Söylediği şey için hangi girdiye dönüp bakacağımızı sorabiliriz.

Bölüm 8 · Difüzyonla metinden görüntü üretimi

Bir model, tariften nasıl kapı resmi çizer?

Usta eski kapıdan söktüğü pirinç tokmağı tezgâha bırakıyor. “Acaba koyu yeşilin üzerinde nasıl durur?” Çırak telefonunu çıkarıp bu kez fotoğraf yüklemeden yazıyor.

> Koyu yeşile boyanmış ahşap bir atölye kapısı çiz. Üzerinde eski bir pirinç tokmak olsun.

Kapının kaç tahtadan yapılacağını söylemedi. Tokmağın biçimini de tarif etmedi. Yine de ortaya çıkan resimde tahtalar belli bir düzende olacak, tokmağın da belli bir biçimi olacak. Tarifin açık bıraktığı bu ayrıntılar nereden gelecek?

Görüntü üretmenin yollarından biri **difüzyon** (*diffusion*). Bu yöntemin ilk bakışta tuhaf bir yanı var. Güzel bir resim üretmesini istediğimiz modeli eğitirken işe elimizdeki resimleri bozarak başlıyoruz. Bunun bize ne kazandırdığını görmek için geçen bölümdeki dört pikseli geri çağıralım.

Resmi biraz bozunca elimizde ne kalıyor?

Üstteki iki pikselin parlaklığı 20, alttakilerin parlaklığı 220 idi. Eğitim programı her piksel için rastgele bir sayı üretsin. Bazıları artı, bazıları eksi olsun. Bunları temiz değerlere ekleyince bazı pikseller biraz açılır, bazıları biraz koyulaşır. Buradaki **gürültü** sözcüğüyle bu rastgele değişimleri kastediyoruz. Hesabı kolay izlemek için tabloda doğrudan toplama yapacağız. Difüzyon eğitiminde görüntünün ve gürültünün karışımdaki payları, seçilen gürültü düzeyine göre ayarlanır.

Piksel	Temiz değer	Programın eklediği gürültü	Modelin göreceği değer
Sol üst	20	+8	28
Sağ üst	20	−3	17
Sol alt	220	−5	215
Sağ alt	220	+6	226

İlk satırı parmağınızla izleyin. Modelin önünde 28 var. Biz hesabı yaptığımız için bu 28'in, 20'ye 8 eklenerek elde edildiğini biliyoruz. Model yalnızca 28'e bakarak aynı sonuca ulaşabilir mi? Başlangıç 25, eklenen sayı 3 olsaydı da önünde 28 olacaktı.

Şimdi öteki pikselleri de açın. Sol üstteki 28'in yanında 17, aşağısında ise 215 var. Görüntünün bütününde koyu bir yüzey ve hemen altında aydınlık bir şerit seçilebiliyorsa 28'in hangi tür bölgeye ait olduğu hakkında bir ipucumuz var. Bu ipucu tam olarak 8 eklendiğini kanıtlamaz. Ama çok sayıda fotoğraf üzerinde komşu değerlerin nasıl değiştiğini öğrenen bir modelin tek başına 28'e bakarak yapabileceğinden daha iyi tahminler yapmasını sağlar.

Eğitimde cevap anahtarını kim yazdı?

İncelediğimiz difüzyon modeline gürültülü görüntüyü, ne ölçüde gürültü karıştırıldığını ve görüntünün yazılı tarifini verelim. Model her konum için oraya eklenen gürültüyü tahmin etsin. Sol üst piksel için tahmini +6 olsun. Programın sakladığı hedef +8 idi. Arada iki birim fark var. Tablomuzda modelin tahminiyle temiz değere dönmeyi deneseydik 28 − 6 = 22 bulacaktık. Gerçek değer olan 20'ye iki birim uzaktayız.

Şimdi tablonun yalnızca ilk satırına bakmak yerine her sütunu yukarıdan aşağıya okuyalım. Her seferinde aynı yolu izliyoruz: sol üst, sağ üst, sol alt, sağ alt. Böylece temiz görüntü **(20, 20, 220, 220)**, eklediğimiz gürültü **(+8, −3, −5, +6)**, modelin gördüğü görüntü de **(28, 17, 215, 226)** vektörü oluyor. Üçü de dört boyutlu. Burada "dört boyut" demek, dört ayrı yerde sakladığımız dört değeri birlikte kullanıyoruz demek. İlk sayı yalnızca ilk sayıyla, ikinci yalnızca ikinciyle toplanıyor. Tabloda satır satır yaptığımız hesabı vektörlerle böyle yazabiliyoruz.

Model de bu örnek için dört gürültü değeri tahmin eder. Diyelim ki tahmini **(+6, −3, −5, +4)** oldu. Hedefle yan yana koyunca ortadaki iki konumda tam isabet ettiğini, ilk ve son konumda ikişer birim yanıldığını görüyoruz. Her tahmini kendi konumundaki hedefle karşılaştırmak önemli. Model ilk piksel için gereken düzeltmeyi son piksele uygularsa toplam parlaklık doğru çıksa bile görüntüdeki açık ve koyu noktalar yanlış yerde kalabilir.

Birinci bölümde parçanın işlenme süresini tahmin ederken doğru cevabı öğrenmek için saate bakmıştık. Burada kimsenin binlerce fotoğraftaki her pikseli tek tek etiketlemesi gerekmiyor. **Gürültüyü programın kendisi ürettiği için tahminle karşılaştıracağı hedef**

zaten elinde. Bütün konumlardaki tahmin hatalarından bir kayıp hesaplanabilir ve modelin ağırlıkları geri yayılımla ayarlanabilir.

> **Meraklısına · Dört pikselden tek bir kayıp nasıl çıkar?**
>
> İlk bölümde sipariş süreleri için yaptığımız hesabı bu kez gürültü tahminlerine uygulayalım. Sol üst piksele +8 eklemiştik, model +6 tahmin etti. Hatayı yine "tahmin eksi hedef" diye yazarsak $6 - 8 = -2$ buluruz. Karesi 4. Ortadaki iki tahmin tam isabetti. Son pikselde ise model +6 yerine +4 tahmin etti. Dört pikselin hata kareleri sırasıyla 4, 0, 0 ve 4. Ortalamalarını aldığımızda bu örneğin kaybı $(4 + 0 + 0 + 4)/4 = 2$ olur.
>
> Şimdi yalnızca ilk tahmini değiştirin. Model +6 yerine +7 deseydi kayıp kaç olurdu? İlk pikselin hata karesi 4'ten 1'e inerdi. Son pikselin payı hâlâ 4 olduğundan yeni ortalama $(1 + 0 + 0 + 4)/4 = 1{,}25$ olurdu. Aynı gürültülü görüntü için iki tahmini böyle karşılaştırabiliyoruz. Eğitimde de ağırlıklardaki küçük bir değişikliğin pek çok örnekte bu ortalamayı hangi yönde etkilediği hesaplanır.

Bir eğitim örneğinde fotoğraftaki kapı kolayca seçilecek kadar az gürültü kullanalım. Bir başkasında görüntü bilgisinin payını azaltıp gürültünün payını artıralım. O zaman kenarlar bile zor seçilir. Modelin gürültü düzeyini de girdi olarak almasının sebebi bu. Hafifçe bozulmuş bir tahta dokusunda bir sonraki düzeltme için kullanılabilecek ipuçlarıyla, tahtaların henüz seçilemediği bir görüntüdeki ipuçları farklıdır.

Ustanın tarifi hesaba nerede giriyor?

Eğitim arşivinde görüntülere eşlik eden açıklamalar olduğunu düşünelim. Bazılarında "pirinç tokmak", bazılarında "siyah demir halka" yazıyor. Metni işleyen bölüm bu sözcükleri vektörlere dönüştürür. Görüntüyü işleyen ağ, gürültü tahminini hesaplarken bu vektörlerden de bilgi alabilir. Yedinci bölümde parçalar arasında izlediğimiz dikkat hesabı, burada görüntünün bir bölgesiyle tarifin ilgili sözcükleri arasında da kurulabilir.

Bunun öğrenmeye nasıl yardım ettiğini görmek için, koyu bir yüzeyin üzerinde güçlükle seçilen küçük bir parlak leke düşünün. Tarifte "eski pirinç tokmak" yazıyorsa o bölgede sarıya çalan metal parıltıları bulunması makul. Model, eğitim boyunca yazıyla görüntü arasındaki bu tür ilişkilerden yararlandığında gürültüyü daha iyi tahmin edebiliyorsa, kayıp hesabı bu ilişkilerin öğrenilmesine katkıda

bulunur. Sonunda ustanın “pirinç” sözcüğü, üretim sırasında hangi renk ve dokuların ortaya çıkacağını etkileyebilir.

Ortada temiz bir resim yokken ilk adım nasıl atılıyor?

Çırak “Çiz”e bastığında program yeni bir rastgele sayı dizisi üretir. Ekrana dökseydik kumlanma gibi görünürdü. Bu kez başlangıçta üzerine gürültü eklediğimiz bir kapı fotoğrafı yok. Öyleyse eğitimde öğrendiğimiz gürültü tahmini burada ne işe yarayacak?

Eğitimde gürültünün ağır bastığı örnekleri hatırlayın. Model böyle bir girdide de tariften ve öğrendiği görüntü düzenlerinden yararlanarak bir tahmin yapmaya çalışmıştı. Şimdi aynı hesabı başlangıçtaki sayılar ve ustanın tarifiyle yapabilir. Üretimi yöneten program, bu tahminle o adımın gürültü düzeyini birlikte kullanarak biraz daha düşük gürültü düzeyine karşılık gelen yeni sayıları hesaplar. **İkinci adımın girdisi, ilk adımın bu sonucudur.** Model tahminini yeni sayılara bakarak yeniden yapar.

Dört piksellik örneğimizde bu adımı iki listeyle izleyebiliriz. Bir listede o anki görüntüyü temsil eden dört sayı, öbüründe modelin o dört konum için tahmin ettiği gürültü var. Program bu iki vektörü ve adımın gürültü düzeyini kullanarak yine dört sayılık bir görüntü vektörü hesaplar. Sonraki turda model bu yeni vektöre bakıp **yeni bir gürültü vektörü** tahmin eder. Eğitimde tahminin yanına koyduğumuz hedef sütununu hatırlayın. Üretimde o sütun yok. Başlangıçtaki rastgele sayıları biliyoruz, ama onlardan ustanın tarifine uygun tek bir kapıya götürecek hazır bir düzeltmeler listemiz yok.

Neden tek seferde bitirmiyoruz? İlk adımda tokmağın çevresini gösterecek belirgin bir kapı kanadı bile yok. Birkaç güncellemeden sonra kapıya benzeyen bir yüzey oluşmaya başlayınca sonraki tahmin artık bu yüzeyin konumundan da yararlanabilir. Aynı tarif her adımda elde, ama tahminin dayandığı görüntü adım adım değişiyor. Bu sırada modelin eğitimde öğrendiği ağırlıklar sabit kalıyor. Güncellenen, üretmekte olduğumuz görüntüyü taşıyan sayılar.

Bazı sistemler bu işlemleri görüntünün daha küçük bir sayısal temsili üzerinde yapar. Buna *latent* ya da *gizil temsil* denir. Son adımda bir çözücü bu temsili ekranda göreceğimiz renkli piksellere dönüştürür. Dört piksellik örneğimizde doğrudan parlaklıkla çalışmak, aynı öğrenme fikrini kâğıt üzerinde izlememizi sağladı.

Gerçek ölçekte “çok boyutlu” ne demek? 512 × 512 piksellik renkli bir görüntüde her piksel için kırmızı, yeşil ve mavi olmak üzere üç değer tutarsak toplam **512 × 512 × 3 = 786.432** sayı eder. Bunları tek listeye dizseydik her görüntü 786.432 bileşenli bir vektör olurdu. Programlar satır, sütun ve renk kanalı düzenini koruyarak bu sayılar üzerinde çalışabilir. Gizil temsille çalışan bir modeldeyse güncellenen sayılar, görüntünün daha küçük bir ızgaraya kodlanmış özelliklerini taşır. Dört piksellik örneğimizden akılda kalacak ilişki her iki ölçekte de geçerli: Tahmin edilen düzeltmenin her bileşeni, o anda güncellenen temsilin bir bileşenine karşılık gelir.

Aynı tariften iki kapı çıkarsa hangisi doğru?

Üç çizimi yan yana asacağımız bir düşünce deneyi yapalım. Üçünde de aynı modeli ve üretim ayarlarını kullanalım. Başlangıç gürültüsünü oluşturmak için program çok sayıda rastgele değere ihtiyaç duyar. Bilgisayarda bunları bir başlangıç sayısından hareketle tekrarlanabilir bir sırayla üretebiliriz. İşte bu başlangıç sayısına **tohum** (*seed*) denir. Tohumu 17 seçersek bütün pikseller 17 ile başlamaz. 17, sayı üretecini belirli bir noktadan başlatır. Üretecin peş peşe verdiği farklı değerler bir araya gelerek ilk gürültü vektörünü oluşturur. Aynı üreteci aynı koşullarda yeniden 17 ile başlatırsak aynı diziyi elde ederiz.

Bunun çırak için pratik bir yararı var. Hem tarifi hem rastgele başlangıcı değiştirirse iki çizim arasındaki farkın hangisinden kaynaklandığını ayırmak güçleşir. Deneyimizde aynı tohumu kullandığımız iki çizimde başlangıç gürültüsünü ve sonraki adımlarda kullanılan rastgele seçimleri tekrarlayabildiğimizi varsayalım.

Çizim	Tarif	Tohum
A	Koyu yeşil ahşap atölye kapısı, eski pirinç tokmak	17
B	Koyu yeşil ahşap atölye kapısı, eski pirinç tokmak	42
C	Koyu yeşil ahşap atölye kapısı, siyah demir halka	17

Önce **A ile B**’yi düşünün. İkisinde de tarif yerine getirilmiş olsun. Birinde geniş üç tahta, diğerinde dar altı tahta görürseniz hangisini hatalı sayarsınız? Usta tahta sayısı vermemişti. Farklı başlangıçlardan yapılan güncellemeler, aynı tarifle uyumlu iki ayrı kapıya ulaşabilir.

Şimdi **A ile C**’ye bakalım. Başlangıç aynı, bu kez tarif değişti. Daha ilk gürültü tahmini farklı bir metinden bilgi alıyor. Sonraki adımlar da bu yeni tahminin etkisini taşıyor. Halkanın malzemesiyle birlikte

kapıya düşen gölgenin veya tahtaların dizilişinin değişmesi bu yüzden mümkün.

Usta elindeki tokmağı çizimin yanına koyuyor. Resimdeki de pirinç, ama ustanınkinin kenarında yıllar önce düşerken oluşmuş küçük bir çentik var. Çırak o çentikten hiç söz etmemişti. Yedinci bölümde fotoğrafa bakıp “Bu ayrıntı nerede görünüyor?” diye sorabiliyorduk. Burada çentiği doğru yerde göstermek isteseydik, ustanın tarifine hangi bilgiyi eklerdik? Belki de en kolayı, tezgâhtaki tokmağın bir fotoğrafını da vermek olurdu.

Bölüm 9 · Referans görüntüyle üretim ve maskeyle düzenleme

Fotoğrafı verince tokmağın çentiği de korunur mu?

Çırak tezgâhtaki pirinç tokmağın fotoğrafını çekiyor. Usta geçen bölümde çizdikleri yeşil kapılardan birini seçmiş. İstediği şey basit görünüyor. “Şu tokmağı bu kapıya takılmış gibi göster. Bir de bizimki olduğu belli olsun.”

Tokmağın dış kenarında küçük bir çentik var. Çırak ilk denemede referans fotoğrafını ve kapının çizimini programa veriyor. Sonuçta pirincin rengi tutmuş, halka doğru büyüklükte, gölgesi de yeşil tahtanın üzerine düşmüş. Usta resmi büyütüyor. “İyi de bizim çentik nerede?”

Bu kez tarifte eksik bilgi arayamayız. Çentik fotoğrafta görünüyor. Peki fotoğrafta bulunan bir ayrıntı sonuca hangi yoldan geçer? Bunu anlamak için çırağın bilgisayarda yapabileceği üç ayrı işlemi izleyelim.

Tokmağı fotoğraftan kesip kapıya yerleştirirsek

Çırak önce sıradan bir resim düzenleyici açıyor. Tokmağın çevresini seçip seçimi kapı resminin üzerine taşıyor. Bunu bir kâğıt fotoğraf ve makasla yaptığınızı düşünün. Yalnız dış kenardan keserseniz halkanın ortasında tezgâhın fotoğrafı kalır. Ortayı da oyduğunuzda boşluktan yeşil kapı görünür. Çırak ekranda aynı özeni gösteriyor. Seçimin sınırını çentiğin içine kadar izliyor ki tokmağı taşıdığında o küçücük girintiden de yeşil tahta görünsün. Tokmağı aynı boyda tutup yalnızca sağa kaydırırsa yüzeyindeki koyu bir leke, çentikle birlikte aynı miktarda sağa gider. İkisi arasındaki uzaklığı kaynak fotoğrafta ve sonuçta piksel piksel ölçebiliriz.

Bu yöntemde başımıza başka bir iş açılıyor. Tezgâhtaki tokmağa ışık sağdan gelmiş, kapı çiziminde ise sol taraf aydınlık. Çırak kesip taşıdığı pirinçteki parlak yansımayı da beraberinde getirdi. Usta çentiği tanıyor ama tokmak yüzeye yapıştırılmış bir etiket gibi görünüyor. Sahnenin ışığına uydurmak için artık bazı pikselleri değiştirmek gerek.

Fotoğraf üretime nasıl yol gösterir?

Şimdi referans görselle çalışabilen bir üretim sistemi düşünelim. Çırak yalnızca “pirinç tokmak” yazdığında pek çok biçim tarife uyuyordu. Fotoğrafta ise başka ipuçları da var. Örneğin bu halkanın alt tarafı üst tarafından daha kalın olsun. Alt kenarı boyunca uzanan parlak şerit de halkanın orada nasıl kavis yaptığını belli ediyor.

Bu örnekte fotoğrafı işleyen parçaya **görüntü kodlayıcısı** deniyor. Kodlayıcı, farklı bölgelerde renklerin, kenarların ve dokuların nasıl bir araya geldiğini sayılarla ifade ediyor. Bu sayıların birlikte taşıdığı bilgiye fotoğrafın *sayısal temsili* diyoruz. “Halkanın alt tarafı kalın” bizim görüntüye bakıp kurduğumuz cümle. Model aynı biçimden yararlanmak için onu önce kelimelere dökmek zorunda kalmıyor. Geçen bölümde model her adımda elindeki gürültülü görüntüye ve yazılı tarife bakarak bir düzeltme tahmin ediyordu. Bu örnekte aynı tahmin, referans fotoğrafından çıkarılan özelliklere de bağlı.

Bunun sonuçta ne fark yarattığını bir soruyla görebiliriz. Kapıdaki tokmağın sağ kenarında hangi pikselin bulunacağını kim belirliyor? Kesip taşırken o pikselin kaynak fotoğraftaki adresini gösterebiliyorduk. Burada o konumun rengi, üretim adımları boyunca hesaplanıyor. Model pirincin kavisli yüzeyini yeni ışığa uygun çizebilir. Aynı hesap, yüzeyin kenarını düzgün bir yay biçiminde de tamamlayabilir.

Çentik özellikle küçükse iki ayrı yerde izini kaybedebiliriz. Diyelim ki giriş fotoğrafı kodlanmadan önce dörtte bir genişliğe küçültülüyor. Fotoğrafta iki piksel genişliğindeki çentik, yeni ölçekte yarım piksel genişliğe karşılık gelir. Yarım pikseli ayrı bir kare olarak gösteremeyiz. Küçültme sırasında çentiğin içindeki koyu renkle yanındaki açık pirinç aynı pikselin rengine katkıda bulunabilir. Küçük resmi yeniden büyüttüğümüzde orada yalnızca soluk bir leke görebiliriz. Girintinin sivri mi, yuvarlak mı olduğunu o renkten çıkarmak güçtür. İkinci olasılıkta ise çentik girişte gayet belirgindir, ama üretim sırasında kapıya çizilen tokmak o ayrıntıyı taşımaz. Yalnız son resme bakarak kaybın hangi aşamada olduğunu söyleyebilir miyiz? İki olasılık da aynı düzgün kenarı verebilir.

Kapının yalnız bir kısmını yeniden çizdirirsek

Usta kapı çiziminin kâğıt çıktısını tezgâha seriyor. Üstüne ince, yarı saydam bir kâğıt koyup eski tokmağın üstünü kalemle tarıyor. “Şuraya bizim tokmağı çizelim. Üst köşedeki budak pek güzel olmuş, oraya dokunmayalım.” Üstteki kâğıdı kaldırınca kalem izleri de onunla birlikte kalkıyor. Tekrar yerine koyduğunda taralı alan, değişecek kısmı tam kapının üzerinde gösteriyor.

Çırak ekranda bunun benzerini yapıyor. Fırçayla eski tokmağın ve gölgesinin üzerinden geçiyor. Program, işaretlediği alanı yarı saydam kırmızıyla gösteriyor. Fırça taşıp budağa değerse çırak o işareti silebilir. İşte hangi yerlerin seçildiğini kaydeden bu ayrı katmana **maske** (*mask*) deniyor. Kırmızı renk çırağın seçimini görmesine yarıyor. “Buraya ne çizilecek?” sorusunun cevabı ise ustanın tarifinden ve verdiği fotoğraftan geliyor.

Peki program bu seçimi nasıl saklar? Kapı resmindeki her küçük renk karesine, yani her piksele, üstteki katmanda bir kutu karşılık gelsin. Örneğimizde fırçanın geçtiği kutulara 1, ötekilere 0 yazalım. Böylece 1 “burayı yeniden çizime aç”, 0 ise “burada eski resmi koru” anlamına geliyor. Model işaretli alanı doldururken çevresindeki tahtadan da yararlanabiliyor. Örneğin yeni tokmak eskisinden küçükse, açığa çıkan aralıkta tahtanın damarını iki yandaki çizgilere uyacak şekilde sürdürebilir. Bir görüntünün seçilen bölgesini çevresinden de yararlanarak tamamlama işlemine *inpainting* deniyor.

Programın yeni çizimden sonra iki resmi bu maskeyle birleştirdiğini varsayalım. Kapının tek bir satırında dört komşu pikselin parlaklığını izleyelim. Büyük sayı daha açık renk olsun.

	1. piksel	2. piksel	3. piksel	4. piksel
Eski kapı	30	34	32	29
Üretilen çizim	35	90	72	24
Maske	0	1	1	0
Birleştirilmiş sonuç	30	90	72	29

İlk piksel için yeni çizimde 35 hesaplanmıştı. Sonuçta 30 kalmasının nedenini gösterebiliyoruz. Birleştirme işlemi orada eski kapıdan okudu. Ortadaki iki pikseldeyse yeni çizimin değerleri kullanıldı. Peki fırçanın kenarını yumuşak seçersek? Sınırdaki bir piksel için “eskiyle yeniyi yarı yarıya karıştır” diyebiliriz. Tablodaki ikinci piksele bunu uygulasaydık eski parlaklık 34 ile yeni parlaklık 90’ın ortalamasını, yani 62’yi görürdük. Gölgenin yeşil tahta üzerinde yavaşça silindiği yerde böyle bir geçiş işe yarar.

Şimdi ustanın iki isteğini bu tabloya yerleştirelim. Kapının üst köşesindeki eski budak maskenin dışında kalıyorsa, örneğimizde budağın korunmasını birleştirme işlemi sağlıyor. Tokmağın çentiği ise yeniden çizilecek alanın içinde. Maske modele çentiğin biçimini söylemiyor. Bunun için hâlâ referans fotoğrafına ve o fotoğrafın üretimde nasıl kullanıldığına bağlıyız.

Çırak son bir deneme yapıyor. Bu kez gerçek tokmağı önce kapının üzerine kendisi yerleştiriyor. Sonra yalnız tokmağın kapıya düşecek

gölgesini ve çevresindeki yeşil tahtayı yeniden çizime açıyor. Usta sonucu büyüttüğünde çentiği buluyor. Peki sağdan gelen eski parlak yansıma? O da yerinde. Çırak, tokmağın yüzeyini korumayı seçtiğinde ikisini birden korumuş oldu.

Usta şimdi yalnız pirincin üzerindeki parlaklığı değiştirmek istiyor. Fırça sizde olsaydı tokmağın neresini maskeye alır, neresini dışarıda bırakırdınız? Sonucu kontrol etmek için de iki ayrı yere bakın. Çentiğin sınırı eski fotoğrafla örtüşüyor mu? Parlaklığın yeni konumu kapının üzerindeki gölgeyle uyuşuyor mu?

Bölüm 10 · Ses algılama, dalga biçimi ve spektrogram

Bir model kapının iki kez çalındığını nasıl fark eder?

Usta pirinç tokmağı yeşil kapıya takıyor. Çırak telefonunda ilk kaydı başlatıyor. Usta tokmağı kaldırıp bırakıyor. Tak! Biraz bekleyip aynı hareketi tekrarlıyor. İkinci vuruşun ardından pirincin ince çınlaması dininceye çırak kaydı durduruyor.

İkinci kayıt için usta tahta tezgâhın başına geçiyor. Çırak yeniden kayıt düğmesine basıyor. Usta bu kez parmağının eklemiyle tezgâha iki kez vuruyor. Vuruşların arasını önceki kayıttaki kadar açıyor. İki kısa, tok ses duyuluyor. Çırak kaydı durdurup telefonu ustaya uzatıyor. Usta kayıtlardan birini seçip çalıyor. “Tokmak,” diyor çırak. “Sonunda hâlâ çınlıyor.” Usta öteki kaydı açıyor. “İkisinde de aynı ritmi vurdum. Peki telefon neyi kaydetti de aradaki farkı şimdi de duyabiliyorsun?”

Bir vuruştan nasıl binlerce sayı çıkar?

Tokmak kapıya çarpınca hem tokmak hem kapı titreşiyor. Yakınlarındaki hava sırayla biraz sıkışıp gevşiyor. Bu küçük basınç değişimleri yayılarak telefonun mikrofonuna ulaşıyor. Mikrofonun içindeki ince zar da bunlarla ileri geri hareket ediyor. Mikrofon bu hareketi değişen bir elektrik sinyaline çeviriyor.

Bu sinyali kâğıda geçirmek sizin işiniz olsaydı bir kalemi soldan sağa sabit hızla yürütüp sinyal yükseldikçe ucunu yukarı, alçaldıkça aşağı oynatabilirdiniz. Telefonun kayıt devresi ise sinyalin o andaki değerini eşit zaman aralıklarıyla ölçüp bir sayı olarak saklayabilir. Her ölçüme bir **örnek**, bir saniyede alınan ölçüm sayısına **örnekleme hızı** denir. Bizim tek mikrofonlu kaydımızda bu hız saniyede 16.000 örnek olsun. Usta bir saniye boyunca tek vuruş da yapsa üç vuruş da yapsa elimizde 16.000 sayı olur. Vuruşları birbirinden ayıran bilgi, o sayıların zaman içinde nasıl değiştiğinde saklı.

O sayılardan birinin 0 olması ne anlatır? Ölçümün tam o anda orta düzeyden sapmadığını. Oysa hızla titreşen zar, ileri giderken de geri dönerken de ortadan geçer. Dolayısıyla tek bir sıfıra bakıp “atölye sessizdi” diyemeyiz. Öncesindeki ve sonrasındaki ölçüler

de sıfıra yakınsa ancak o zaman kaydın o kısmının sakin geçtiğini söyleyebiliriz.

Çırak ekrandaki kaydı büyütüyor. Her sayıyı ölçüldüğü ana bir nokta olarak koyduğumuzu düşünün. Noktaları soldan sağa izlediğimizde ortaya çıkan şekle **dalga biçimi** deniyor. Çizginin orta düzeyin iki yanına gidip gelmesi basınçtaki artış ve azalışları gösteriyor. Tek bir tokmak vuruşu bile çizgiyi defalarca yukarı aşağı götürebilir. Çırak her tepeyi bir vuruş saysaydı usta kapıda epey beklemiş görünürdü.

Bunu bir cetvelle deneyebilirsiniz. Cetvelin bir ucunu masaya bastırıp dışarı taşan ucunu aşağı çekin ve bırakın. Elinizi bir kez çektiniz, ama uç defalarca aşağı yukarı gidip geldi. Tokmağın çarpması da titreşimi böyle başlatır. Kaydı çok yakından incelerken bir vuruşun içindeki salınımları, daha geniş bir zaman aralığına bakarken bu salınımların topluca nasıl başlayıp söndüğünü görürüz.

İki vuruşu nerede görüyoruz?

Şimdi ekrandan biraz uzaklaşalım. Çırak kaydın iki yerinde çizginin birden geniş bir alana yayıldığını, ardından yeniden inceldiğini görüyor. Yakından gördüğümüz hızlı titreşimler, uzaktan bakınca bu iki kabarıklığın içine sığıyor. İlk kabarıklık kaydın 0,20’nci saniyesinde, ikincisi 0,70’inci saniyesinde başlıyorsa usta ilk vuruştan yarım saniye sonra yeniden vurmuş demektir.

> **Meraklısına · İki vuruşun arasını örneklerden nasıl ölçeriz?**
>
> Çırak ekranda saniyeler yerine örnek numaralarını görseydi aynı aralığı bulabilir miydi? Kaydın başındaki örneğe 0 numarasını verelim. Saniyede 16.000 örnek alındığına göre ilk vuruşun başladığı 0,20 saniyeye $0{,}20 \times 16.000 = 3.200$ numaralı örnek denk gelir. İkinci vuruş 0,70 saniyede, yani 11.200 numaralı örnekte başlar. İki başlangıç arasında $11.200 - 3.200 = 8.000$ örnek aralığı vardır. Bunu saniyeye çevirmek için örnekleme hızına böleriz. Böylece $8.000 / 16.000 = 0{,}50$ saniye buluruz.
>
> İkinci bir telefon aynı iki vuruşu saniyede 32.000 örnek alarak kaydetseydi ne değişirdi? Bu kez iki başlangıç arasında 16.000 örnek aralığı sayardık. Geçen süre yine $16.000 / 32.000 = 0{,}50$ saniye olurdu. Ustanın ritmi aynı kaldı. İkinci telefon o yarım saniye içinde iki kat daha sık ölçüm aldı.

Bir program bu değişimi nasıl yakalayabilir? Kaydı kısa zaman dilimlerine bölelim. Her dilimde ölçümlerin orta çizgiden ortalama ne kadar uzaklaştığına bakalım. Yukarı doğru dört birimle aşağı doğru dört birim, uzaklık hesabında aynı değeri verir. Böylece kuvvetli bir titreşimin iki yönü birbirini silmez. Elde ettiğimiz sayı, o kısa süre içinde titreşimin ne kadar güçlü olduğuna dair bir özet olur.

Dört ölçümlük küçük bir hesap yapalım. Vuruş sırasında art arda +4, −4, +6 ve −6 okumuş olalım. Toplarsak sıfır çıkar. Oysa ölçümlerin orta düzeyden ne kadar saptığını merak ediyoruz. Orta çizgiye uzaklıklar 4, 4, 6 ve 6. Toplamı 20, ortalaması 5. Sessiz bir aralıkta +1, 0, −1 ve 0 okusaydık ortalama uzaklık 0,5 olurdu. Program aynı hesabı kaydın peş peşe gelen dilimlerine uyguladığında titreşimin nerede güçlendiğini izleyebilir. Örneğin saniyede 16.000 ölçüm alırken saniyenin yüzde biri kadar bir dilime bile 160 ölçüm sığar.

Bizim temiz kayıtta bu özet önce küçük, sonra birden büyük, biraz sonra yeniden küçük çıkar. Yarım saniye sonra benzer bir yükseliş daha görürüz. İki başlangıç bulduk. Peki usta ilk kez tokmağa, ikinci kez tezgâha vursaydı? Yine iki yükseliş bulabilirdik. Çırak yalnız zamanlamaya baksaydı ikinci sesin neden farklı geldiğini açıklayamazdı.

Çınlama kaydın neresinde?

Masaya parmağınızla bir kez vurup ardından bir bardağın kenarına hafifçe dokunduğunuzu düşünün. İkinci seste, parmağınızı çektikten sonra bile kulağınızın izleyebildiği ince bir ton kalabilir. Atölyedeki iki kaydı ayırırken çırağın dikkatini çeken de böyle bir ayrıntıydı. Vuruşun başlangıcından sonra hangi titreşimler ne kadar süre duyuluyor?

Bir titreşim saniyede yüz kez tekrarlanıyorsa **frekansı** 100 hertzdir. Saniyede bin kez tekrarlanan bir titreşim daha yüksek frekanslıdır ve tek başına dinlendiğinde daha ince duyulur. Tokmağa vurduğumuzda aynı anda farklı hızlarda pek çok titreşim oluşur. Mikrofona bunların birleşimi ulaşır. Üstelik kimi titreşimler hızla sönerken kimileri biraz daha sürer.

Burada iki ayrı hızla karşılaştık. Saniyede 16.000, telefonun kaç kez ölçüm aldığıydı. 100 hertz ise ölçtüğümüz titreşimin bir saniyede kaç tam çevrim yaptığı. Yalnızca bu 100 hertzlik ton çalarken telefon bir çevrim boyunca 160 kez ölçüm alır. Aynı telefonla 1.000 hertzlik bir ton kaydedince çevrim başına 16 ölçüm düşer. Çırak ince sesi

kaydederken telefonun saatini hızlandırmadı. Aynı aralığa daha çok titreşim sığdı.

Çırak kareli bir kâğıt alıyor. Her sütun kaydın kısa bir zaman dilimini göstersin. Alt sıralara yavaş titreşimleri, yukarı çıktıkça daha hızlı titreşimleri yerleştirsin. Program her dilimde hangi frekans çevresinde ne kadar enerji bulunduğunu hesaplasın. Çırak da enerjisi yüksek kutuları koyuya boyasın. Böyle bir gösterime **spektrogram** deniyor. Sayfada sağa giderken sesin zaman içinde nasıl değiştiğini, tek bir sütunu aşağıdan yukarı okurken o anda hangi titreşimlerin baskın olduğunu izleyebiliriz.

Peki tek mikrofondan gelen sayılarla aynı sütunda birkaç sırayı nasıl boyayacağız? Program kısa bir ses parçasını ele alıp içinde değişik hızlarda düzenli salınımların ne kadar payı olduğunu hesaplayabilir. Bunu, aynı anda basılan iki piyano tuşunu dinlemeye benzetebiliriz. Kulağımıza tek bir hava basıncı dalgası ulaşır ama içinde hem kalın hem ince notayı duyabiliriz. Program da o kısa parçadaki yavaş ve hızlı salınımlar için ayrı değerler çıkarır. Sıradaki kısa parçaya geçince yeni bir sütun elde ederiz.

Bizim tokmak kaydımızda ilk darbe anında birçok sıra koyulaşabilir. Sonraki sütunlarda bunların çoğu açılırken üst taraftaki birkaç kutu bir süre daha koyu kalabilir. İşte o yatay izin karşılığını çırak ince bir çınlama olarak duymuştu. Tezgâh kaydında aynı zaman aralığına bakınca izin daha erken silindiğini düşünelim. İki kaydın vuruş aralığı aynı kaldığı hâlde sayfalar artık farklı görünüyor.

O izi görünce modele kim “tokmak” diyor?

Çırak küçük bir deney hazırlıyor. Farklı uzaklıklardan, kimi sert kimi hafif çok sayıda vuruş kaydediyor. Her kayıtta neye vurduğunu bildiği için yanına “tokmak” veya “tezgâh” yazabiliyor. Bu yazı eğitim sırasında kullanılacak **etiket**. Spektrogramla çalışan bir modele önce kaydın sayısal tablosu veriliyor. Model iki etiket için birer puan hesaplıyor. Bir tezgâh kaydında tokmağa yüksek puan verirse tahminiyle çırağın koyduğu etiket arasındaki hata, ağırlıklarını ayarlamakta kullanılıyor. İlk bölümdeki öğrenme fikri burada da iş başında.

Çırak ilk gün bütün tokmak kayıtlarını yakından, bütün tezgâh kayıtlarını da odanın öteki ucundan alsaydı ne olurdu? Model, güçlü gelen seslere “tokmak” diyerek pek çok örnekte doğru cevaba ulaşabilirdi. Verilen etiket hangi nesneye vurulduğunu bildiriyor, modelin o cevaba giderken hangi ipucunu kullanacağını söylemiyor. Bu yüzden çırak her iki sesi de hem yakından hem

uzaktan kaydediyor. Sonra eğitimde hiç kullanmadığı birkaç kayıt açıp yanıtları kontrol ediyor. Özellikle uzaktan kaydedilmiş tokmakla yakından kaydedilmiş tezgâh, sesin gücüne fazla bel bağlayan bir modeli ele verebilir.

Usta deneyi merak ediyor. Pirinç halkayı eliyle tutup kapıya iki kez vuruyor. Elinin teması titreşimi çabuk söndürüyor. Çırak yine iki vuruş sayabiliyor ama bu defa alıştığı çınlama yok. Model eğitimde hep serbestçe çınlayan tokmaklar duymuşsa bu yeni kayda “tezgâh” diyebilir. Bunu sınamak için usta elini çekip bir kayıt daha yapıyor. Aynı tokmak, aynı kapı, aynı ritim. İki tahmini yan yana koymak, çınlamanın model için ne kadar önemli olduğuna dair bir ipucu verebilir.

Çırak kaydı açık unutmuş. Sonunda ustanın sesi duyuluyor. “Kapı açık.” Telefon bu iki kelimeyi de aynı yoldan, peş peşe gelen sayılar hâlinde kaydetti. Çırak kendi sesiyle “Kapı açık” dediğinde sayıların çoğu değişecek. Peki iki kaydı yazıya döken bir model, ikisinde de aynı kelimeleri nasıl bulacak?

Bölüm 11 · Konuşma tanıma ve metinden konuşma üretimi

Aynı cümleyi iki kişi söyleyince ne değişir?

Çırak dün açık unuttuğu kaydın sonunu ustaya dinletiyor. Hoparlörden ustanın sesi geliyor. “Kapı açık.” Çırak yeni bir kayıt başlatıp aynı sözleri söylüyor. İki kaydı da telefonda yazıya döktürüyor. Ekranda alt alta iki satır var. İkisinde de “Kapı açık.” yazıyor.

Usta telefonun sesini kapatıp satırlardan birini gösteriyor. “Hangimiz söylemişiz?” Çırak gülüyor. Sırasını hatırlamasa bilemeyecek. Oysa az önce iki sesi de ilk kelimeden tanımıştı. Demek ki kayıttan yazıya geçerken bazı ayrıntılar geride kaldı. Önce aynı yazıya nasıl ulaştığımıza bakalım. Sonra yalnız o yazıyı verip telefondan konuşmasını isteyeceğiz.

Kelimenin içinde sabit kalan ne var?

Usta “kapı” kelimesini ağır ağır söylüyor. Çırak hemen ardından aceleyle tekrarlıyor. Şimdi sıra sizde. “Kapı” derken dudaklarınızın ne zaman birbirine değdiğine dikkat edin. “A” sesini çıkarırken ağzınız açık. “P”ye gelirken dudaklarınız birleşiyor. Sonra yeniden açılıyor ve “ı”yı duyuyoruz. Kelimeyi uzatarak bir daha söyleyin. “A”ya daha çok zaman ayırabilirsiniz. Dudaklarınızı kapatıp açtığınız yer yine iki ünlünün arasında kalır.

Telefonun mikrofonu dudaklarımızı göremiyor. Ama kapanmanın ve yeniden açılmanın seste bıraktığı izi yakalayabiliyor. Geçen bölümde kısa zaman dilimlerini yan yana dizmiş, her dilimde hangi titreşimlerin güçlü olduğunu gösteren bir tablo elde etmiştik. Aynı spektrogramı burada kullanalım. “A” sürerken bazı frekans bölgelerinde belirgin enerji görüyoruz. Dudaklar kapanınca bu enerji azalıyor. Açılışta kısa bir hava boşalması, ardından da “ı”nın kendine özgü dağılımı geliyor. Çırağın daha hızlı konuştuğu kayıtta bu değişimler daha az sütuna sığıyor.

İki tabloyu üst üste koyup aynı karelerin koyulaşmasını bekleseydik güçlük çekerdik. Hem süreler değişti hem de ustayla çırağın sesleri farklı. Buna karşılık **hangi değişimin hangisinden sonra geldiği** iki kayıtta da bir ipucu. Tek başına dudak kapanması “p”yi tanımaya yetmez. Örneğin “m” derken de dudaklarımız kapanır.

“Ama” ve “apa” diye sesli söyleyin. “M” sırasında burnunuzdan sürdürebildiğiniz uğultuyu dinleyin. Birbirine benzeyen iki hareketi ayırmak için kapanmadan önce, kapanma sırasında ve hemen sonrasında duyulanlar birlikte işe yarar.

Sesin neresine boşluk koyacağız?

Çırak bu kez “Kapı açık” derken iki kelimenin arasında özellikle bekliyor. Telefonda arayı görmek kolay. Sonra aynı cümleyi, kapıda bekleyen bir arkadaşına seslenir gibi tek solukta söylüyor. Yazıda yine bir boşluk var. Kayıtta o boşluğa denk gelen tertemiz bir sessizlik bulmak bu kez zor. Üstelik az önce tek kelimenin ortasında, “p”yi söylerken de sesin kısa süreliğine zayıfladığını gördük. Her sakin yerde bir kelimeyi bitirsek “kapı”nın kendisini de yanlış yerden bölebiliriz.

Bir **konuşma tanıma** modelinin nasıl öğrenebileceğini düşünelim. Elimizde pek çok kişiden alınmış konuşma kayıtları ve her birinin yanında söylenenleri yazıya döken bir satır olsun. Eğitimde model sesin kısa dilimlerini işler. Yan yana gelen dilimlerden daha uzun ses örüntüleri çıkarır ve bunlara bakarak bir yazı dizisi tahmin eder. Yazı küçük parçalar hâlinde üretilebilir. Bu parçaların her biri, önceki bölümlerde tanıştığımız bir *token* olabilir. Bir token birkaç harfi kapsayabilir ve onu destekleyen ses kayıtta pek çok zaman dilimine yayılabilir.

Örneğin modelin bir aşamada “Kapı aç” yazdığını düşünün. Kaydın sonunda “ık” sesleri de duyuluyor. Eğitim örneğinin yazısında “açık” var. Modelin son parçaya düşük olasılık vermesi eğitim hatasına katkı yapar. Ağırlıklar ayarlanırken bu son sesleri de hesaba katan tahminler güçlenebilir. Başka bir kayıtta birisi gerçekten “Kapı aç” derse bu kez doğru sonuç orada durmaktır. Pek çok kayıt ve yazı çifti üzerinde çalışmak, modele hem sesin nerede sürdüğünü hem de hangi yazı parçalarının o seslerle birlikte geldiğini öğrenme fırsatı verir. Bu tür bir sistemde çırağın oturup her harfin başladığı saniyeyi tek tek işaretlemesi gerekmez.

Çırak bu fikri sınamak için üç küçük kayıt hazırlıyor. İlkinde usta “kapı” diyor. İkincisinde çırak aynı kelimeyi daha ince bir sesle söylüyor. Üçüncüsünde yine çırak konuşuyor ama bu kez “küpü” diyor. Yazıya dökerken birinci kayıttan ikinciye geçtiğimizde “kapı”yı korumak, ikinciden üçüncüye geçtiğimizde ise iki ünlüyü de değiştirmek istiyoruz. Üç kayıtta da dudaklar “p” için kapanıyor. Öyleyse biraz önce bulduğumuz o ipucu tek başına kelimeyi seçmeye yetmiyor.

Siz de “ka” ve “kü” hecelerini aynı notada söylemeyi deneyin. Sesinizin perdesini sabit tutsanız bile “ü”ye geçerken dudaklarınızı yuvarladığınızı hissedersiniz. Ağzın biçimi değişince içerideki titreşimlerin bazıları daha çok güçlenir. Spektrogramda enerjinin toplandığı bölgeler de yer değiştirir. Şimdi de “ka”yı önce kalın, sonra ince bir sesle söyleyin. Bu kez perde değiştiği hâlde ikisinde de “a” duyulabilir. Farklı konuşmacılardan gelen eğitim örnekleri, modele hangi değişimlerle birlikte yazının da değiştiğini öğrenme fırsatı verir. Böylece hızlı bir “kapı” ile yavaş bir “kapı” aynı yazıya ulaşabilirken, aynı kişinin aynı hızda söylediği “küpü” ayrı bir sonuç verebilir.

Peki usta ikinci kelimeyi söylerken tezgâha bir çekiç düşerse? Başlangıçtaki “Kapı” belirgin, devamı gürültünün altında kalmış olabilir. Model, eğitimde karşılaştığı kelime dizilerinden de yararlanarak “açık” yazabilir. Çırak ekrandaki düzgün cümleyi okuyunca son hecenin kayıtta ne kadar zor duyulduğunu fark etmeyebilir. Bunu kendi kulağınızla da sınayabilirsiniz. Bir arkadaşınıza kaydı yalnız son hecesiyle dinletin. Sonra baştan dinletin. Önceki kelimeleri duyunca aynı belirsiz ses için daha güçlü bir tahmini olabilir.

Şimdi telefona yalnızca yazıyı verelim

Çırak yeni bir sayfaya “Kapı açık.” yazıyor ve **sesli oku** düğmesine basıyor. Bu kez programa ne ustanın ne de çırağın kaydını vermiş. Hoparlörden yine de bir ses geliyor. “Açık” kelimesi belli bir hızda söyleniyor, ses cümlenin sonuna doğru alçalıyor. Usta sayfaya bakıyor. “Burada onu ne kadar uzatacağı yazıyor mu?”

Siz de iki türlü okuyun. Önce sokaktan “Girebilir miyim?” diye seslenen birine cevap verin. Sonra sabah atölyeye geldiğinizi ve dün kilitlediğiniz kapıyı aralık bulduğunuzu düşünün. Yine aynı iki kelimeyi söyleyin. İkinci okuyuşta “açık” üzerinde biraz daha uzun durmuş olabilirsiniz. Bir hecenin ne kadar süreceği, sesin nerede yükselip alçalacağı ve vurgunun nereye düşeceği birlikte **prozodi** adını alır. Noktalama bir ipucu verir. Yine de aynı noktayla biten iki okuyuş farklı duyulabilir.

Bir **konuşma sentezi** sisteminin izleyebileceği yollardan biri şöyle. İlk parça yazıdaki seslerin hangi sırayla söyleneceğini ve bunların zaman içinde nasıl duyulabileceğini tahmin eder. Çıktısını geçen bölümün kareli kâğıdına benzetebiliriz. Bu kez kâğıdı eldeki kaydı çözümleyerek doldurmuyoruz. Henüz söylenmemiş bir cümle için hangi anda hangi frekans bölgelerinin güçleneceğini tahmin ediyoruz. Örneğin “a” uzatılacaksa ona ait dağılım daha çok sütun boyunca

sürer. Sisteme seçilmiş bir konuşmacı sesi veya bir okuma tarzı da verilmişse bu seçimler tahmini etkileyebilir.

Peki sistem yazılı satırdan bir heceye kaç sütun ayıracağını nasıl öğrenebilir? Bir eğitim kaydında konuşmacının “a”yı uzattığını düşünelim. O kayıttan hesaplanan tabloda “a”nın izi geniş bir yer kaplar. Yazıdan üretilen tabloda model “p”ye çok erken geçerse, gerçek kayıtta “a” hâlâ sürerken tahminde dudakların kapanmasına karşılık gelen değişim başlar. İki tablo arasındaki bu fark, modelin ağırlıklarını ayarlamak için kullanılabilir. Çok sayıda örnekte kelimeler, noktalama ve söyleyiş birlikte değiştikçe model farklı metinlere uygun süre ve vurgu örüntüleri öğrenebilir.

Usta telefonda aynı cümleyi iki ayrı okuma sesi seçerek dinliyor. İki okuyuşun hızını eşitlediğinde bile sesleri ayırt edebiliyor. Buna gündelik dilde **ses rengi**, müzikte de **tını** deriz. İki kişi aynı notada aynı ünlüyü uzatırken bile, temel titreşime eşlik eden daha yüksek frekanslı titreşimlerin güçleri farklı olabilir. Birden çok sesle eğitilen bir sentez sisteminde her kaydın hangi konuşmacıya ait olduğu biliniyorsa, model o konuşmacıyı belirten bir girdiyi de kullanmayı öğrenebilir. Çırak menüden başka bir ses seçtiğinde “Kapı açık” yazısı yerinde durur ama tahmin edilen tablodaki enerji dağılımı değişebilir. “Açık” kelimesini uzatmak ise başka bir değişikliktir. Okur bunu tek başına da deneyebilir: Kendi sesiyle aynı kelimeyi bir kez kısa, bir kez şaşkınlıkla uzatarak söylemesi yeter.

İkinci parçanın işi bu tablodan, hoparlörü sürebilecek bir ses dalgası üretmek. Bunu yapan bileşene *vocoder* denir. Kareli kâğıtta kısa bir zaman diliminin özeti vardı. Hoparlörü oynatmak içinse o aralık boyunca peş peşe gelecek tek tek dalga değerlerine ihtiyaç var. Eğitimde gerçek ses kayıtlarıyla onların tablolarını birlikte gören bir vocoder, verilen tabloya uygun bir dalga üretmeyi öğrenebilir. Telefon bu değerleri elektrik sinyaline çevirip hoparlöre gönderdiğinde hava yeniden titreşmeye başlar.

Usta son bir deneme istiyor. Çırak ustanın kendi kaydını **oynat** düğmesiyle açıyor. O sırada arkadan gelen hafif sandalye gıcırtısı bir kez daha duyuluyor. Ardından çırak aynı kayıttan çıkarılan “Kapı açık.” satırına dokunup **sesli oku** diyor. Kelimeler geri geliyor. Peki gıcırtı hangi anda gelecekti? Çırağın elinde yalnızca o satır kalsaydı, sandalyenin sesini ikinci kelimeden önce mi sonra mı duyduğunu oradan öğrenebilir miydi?

Bölüm 12 · Videoda hareket algılama ve video üretimi

Kapı açılıyor mu, kapanıyor mu?

Çırak telefonu tezgâha dayıyor. Yeşil kapı kapalı, telefonun kamerası da tam karşısında. Çırak kaydı başlatınca usta kapıyı yavaşça aralıyor. Çırak iki saniye sonra kaydı durduruyor. Videonun ortasından bir görüntü seçip ustaya gösteriyor. Kapıyla kasa arasında bir karış boşluk var. “Çekimi sen hatırlıyorsun,” diyor. “Yalnız bu resmi gören biri kapının hangi yöne gittiğini bilebilir mi?”

Çırak bu kez filmin tamamını sessiz oynatıyor. Aralık gittikçe büyüyor. Ardından aynı kaydı sondan başa oynatıyor. Şimdi de kapı kapanıyor. Ortada durdurduklarında yine aynı bir karış boşluğa bakıyorlar. Tek resim, “Kapı ne kadar açık?” sorusuna ipucu veriyordu. “Ne tarafa hareket ediyor?” sorusunu ise o resmin hemen öncesini ve sonrasını görünce cevaplayabildiler.

Hareket resimlerin neresinde?

Bir defterin sağ alt köşesine küçük bir kapı çizin. Sonraki sayfada kapıyı biraz daha aralık, bir sonrakinde biraz daha açık çizin. Başparmağınızı sayfaların kenarında gezdirdiğinizde kapı hareket eder. Bir sayfayı koparıp masaya bırakırsanız kâğıdın üzerinde hâlâ kapının o andaki biçimi vardır. Hareketin yönünü görebilmek için öteki sayfalarla ilişkisini de bilmeniz gerekir.

Videodaki bu tek görüntülerin her birine **kare** denir. Çırağın telefonu saniyede 25 kare kaydediyorsa, iki saniyelik çekimde 50 kare vardır. Art arda gelen iki karenin kayıt anları arasında saniyenin 25’te biri, yani 40 milisaniye geçer. Aynı 50 kareyi saniyede 50 kare hızında göstersek usta ekranda kapıyı bir saniyede açar. Karelerin sırası hareketin yönünü, gösterim aralığı ise onu ne kadar hızlı gördüğümüzü etkiler.

Şimdi bu 50 kareden eşit aralıklarla üçünü seçelim. Telefon sabit. Kapının üstüne yakın bir yükseklikte, kapının serbest kenarıyla kasa arasında görünen aydınlık şeridin genişliğini resim üzerinde ölçelim. Aşağıdaki sayılar bu çekimi kâğıt üzerinde izleyebilmek için seçilmiş örnek değerler.

Gösterim sırası	İlk görüntü	İkinci görüntü	Üçüncü görüntü
İleri oynatırken şeridin genişliği	4 piksel	10 piksel	16 piksel
Geri oynatırken şeridin genişliği	16 piksel	10 piksel	4 piksel

Üç değerin ortalamasını alırsanız iki satırda da 10 çıkar. Oysa ilk satırda her geçişte 6 piksel kazanıyor, ikincide 6 piksel kaybediyoruz. Bir sisteme yalnızca ortalamayı verirsek yönü bulması için gereken bilgiyi kendi elimizle silmiş oluruz. İkinci ölçümden birinciyi çıkarmak kadar küçük bir işlem bile sırayı hesaba kattığı için bu örnekte işe yarar.

> **Meraklısına · Şerit hangi yönde, ne kadar hızlı değişiyor?**
>
> Çırak tablodaki üç görüntüyü kaydın 0,40'ıncı, 0,60'ıncı ve 0,80'inci saniyelerinden seçmiş olsun. İlk iki görüntü arasında şerit 4 pikselden 10 piksele genişliyor. Değişim $10 - 4 = +6$ piksel. Aradan 0,20 saniye geçtiğine göre şeridin ortalama genişleme hızı saniyede $6 / 0{,}20 = 30$ piksel. Bir sonraki aralıkta da aynı sürede 10 pikselden 16 piksele çıkıyoruz.
>
> Şimdi kaydı aynı hızda geriye oynatın. İlk geçişte genişlik 16 pikselden 10 piksele düşüyor. Bu kez $10 - 16 = -6$ piksel buluyoruz. Ekranda yine 0,20 saniye geçtiği için sonuç saniyede -30 piksel. Artı işareti şeridin genişlediğini, eksi işareti daraldığını gösteriyor. Ölçtüğümüz hız görüntüdeki aydınlık şeride ait. Kapının gerçekte kaç derece döndüğünü bulmak için kamera ile kapının konumu hakkında ek bilgi gerekir.

Model hangi kenarı izleyeceğini nasıl bulur?

Çırak hangi şeridi ölçeceğini biliyordu. Modele verdiğimiz görüntülerde ise "kapının kenarı burada" yazan bir ok yok. Üstelik kapı açılınca pirinç tokmak ekranda yer değiştiriyor. İlk karede tokmağın bulunduğu noktaya ikinci karede bakarsak orada dümdüz yeşil

tahta görebiliriz. Aynı noktaya tekrar bakmak, aynı parçayı takip etmeye yetmiyor.

Bir video modelinin kullanabileceği yollardan birini düşünelim. Model her kareyi küçük bölgelere ayırıyor. Her bölgedeki renk, kenar ve doku ilişkilerini bir grup sayıyla temsil ediyor. Yedinci bölümde tek görüntü için kullandığımız fikir burada da geçerli. Bu kez her bölgenin görüntüdeki yerine ek olarak **hangi kareden geldiği** de hesaba katılıyor. Böylece model, önceki karede halkanın göründüğü bölgeyle sonraki karede biraz yana kaymış benzer bir bölge arasında ilişki kurabilir. Kapı kasasının iki karede de aynı yerde kalması, halkanın ona göre nasıl hareket ettiğini yorumlamaya yardım eder.

Böyle ilişkilerin hangisinin işe yaradığını model nasıl öğrenir? Çırağın, bazıları açılan, bazıları kapanan kapıları gösteren kısa videolar topladığını düşünelim. Her videonun yanında hareketin adını yazıyor. Modelin tahmini bu adla uyuşmadığında eğitimde ağırlıkları ayarlanıyor. Çırak aynı çekimin hem ileri hem geri oynatılmış hâlini doğru adlarla eğitime katarsa yeşil boya, tokmağın biçimi ve arka duvar iki örnekte de aynı kalır. Buna rağmen beklenen cevap değişir. Bu örnek çifti, kareler arasındaki değişimi hesaba katmanın neden işe yaradığını gösterir. Modelin bunu gerçekten öğrenip öğrenmediğini görmek için başka bir kapının çekimini de iki yönde oynatıp tahminlerini karşılaştırabiliriz.

Çırak bir deneme daha yapıyor. Usta bu kez bir saniye içinde kapıyı açıp yeniden kapatıyor. Modelin incelemesi için yalnız baştaki ve sondaki kareyi seçersek ikisinde de kapı kapalı görünür. Model bu iki kareye dayanarak arada ne olduğunu kesin olarak söyleyebilir mi? Ortadan bir kare daha verdiğimizde kapıyı aralık görüyorsa eline yeni bir kanıt geçer. Burada telefonun saniyede kaç kare **kaydettiği** ile programın bunlardan hangilerini **incelediği** iki ayrı karardır. Çırağın elinde hareketin tamamı varken modele verdiği görüntülerde o hareket kaybolabilir.

Aynı kapıyı elli kez çizersek film olur mu?

Usta bu kez kapıyı kapatıp tezgâha oturuyor. Çırak programa “Yeşil atölye kapısı yavaşça açılıyor” yazıyor. Sekizinci bölümde bu tariften bir kapı resmi üretmeyi konuşmuştuk. Çırak her kareyi ayrı ayrı üretse ve her seferinde kapıyı biraz daha açık istese ne olurdu? Elli resmi art arda gösterebilirdi. Gelgelelim üçüncü resimde tahtanın budağı tokmağın üstünde, dördüncüde altında belirebilirdi.

Durdurup baktığınızda iki resim de gayet makul görünür. Oynatınca budak sıçrar.

Sekizinci bölümdeki gürültüden resim üretme yöntemini kısa bir videoya uyarlayalım. Bu kez modelin önündeki sayılar birkaç karelik bir parçayı temsil etsin. Eğitim sırasında gerçek bir çekimin bu temsiline miktarını bildiğimiz rastgele gürültü ekleyebiliriz. Modelden eklenen gürültüyü tahmin etmesini istediğimizde, bir karedeki belirsiz bölgeyi yorumlamak için komşu karelerden de yararlanabilir. Bir karede tokmak bulanıklaşmışsa hemen önceki ve sonraki karedeki konumu, o aralıkta halkanın nereden geçmiş olabileceğine ipucu verir. Model pek çok çekim üzerinde çalıştıkça biçimlerin hareket ederken nasıl değiştiğine ilişkin düzenlilikleri öğrenebilir.

Çırak tarifteki “yavaşça” kelimesinin izini sürüyor. Eğitimde kısa filmlerin yanında hareketi anlatan yazılar da bulunduğunu düşünelim. İki örnekte de kapı başlangıçta kapalı, sonunda bir karış açık olsun. Birinde kapı bu yolu iki saniyeye yayarak alıyor. Ötekinde hemen açılıp bekliyor. İlk ve son kareler birbirine benzese bile aradaki kareler farklı. Model, düzeltme tahminini yaparken yazıdan da yararlanıyorsa “yavaşça açılıyor” sözleriyle birlikte nasıl bir değişimin görüldüğünü bu tür örneklerden öğrenebilir. Çırağın her kare için kapının açısını derece derece yazması gerekmez.

Üretim sırasında elimizde o gerçek çekim yok. Başlangıçta rastgele sayılar ve çırağın tarifi var. Model her adımda birkaç karenin o anki hâlini birlikte hesaba katarak düzeltmeler tahmin eder. Bir adımı tamamladığımızda filmin yalnız ilk karesini bitirmiş olmayız. Bütün kısa parçanın sayısal temsili biraz daha şekillenmiştir. Bu, video üretmenin yollarından biridir. Karelerin birbirinden bilgi alabilmesi, kapı dönerken tokmağın onunla birlikte dönmesine yardım eder.

Burada iki ayrı sırayı gözümüzde canlandıralım. Usta masaya yan yana üç kâğıt koyuyor. Soldaki filmin başını, ortadaki ortasını, sağdaki sonunu temsil ediyor. Çırak bir düzeltme turunda üç kâğıtta da çalışıyor. Sonraki turda yine aynı üç kâğıda dönüyor. **Masada soldan sağa gitmek filmin içinde ilerlemek, aynı kâğıda bir tur sonra dönmek ise o anın çizimini geliştirmek.** Ekranda izlediğimiz hareket, iş bittiğinde soldan sağa sıralanmış görüntülerden doğuyor.

Şimdi ilk kâğıdın ustanın çektiği kapalı kapı fotoğrafı olduğunu düşünelim. Usta bu fotoğrafı başlangıç karesi olarak verip kapının aralanmasını istiyor. Kapı kapalıyken arkasındaki atölyeden tek bir raf bile görünmüyor. Ortadaki karede ince bir aralıktan mavi bir rafın ucu belirirse son karede ne görmeyi beklersiniz? Kapı

biraz daha açıldığında rafın devamı görünebilir. Son karede aynı yerde birden kırmızı bir perde belirirse gözümüz bu değişikliğe takılır. Modelin iki işi var. İlk fotoğrafta görünmeyen yer için bir ayrıntı üretmesi, sonra da aralık genişledikçe o ayrıntıyı sürdürmesi gerekiyor. Raf göründüğü karelerde kusursuz çizilse bile usta "Benim atölyemde öyle bir raf yok" diyebilir. Fotoğraf, kapının arkasında gerçekte ne bulunduğunu modele göstermemişti.

Çırak bu kez dokuzuncu bölümdeki çentikli tokmağın fotoğrafını da veriyor. Usta üretilen filmi durduruyor. İlk karede çentik halkanın dış kenarında. Kapı döndükçe halka daha dar görünüyor ve çentik de ekranda yana kayıyor. Usta son kareyi büyütüyor. Çentiği ararken parmağını ilk karede dokunduğu **ekran noktasına** mı koymalı, yoksa dönmüş halkanın **aynı kenarını** mı bulmalı? Halkanın o kısmı artık görünmüyorsa çentiğin de gözden kaybolması beklenir. Ama hâlâ karşımızdaysa ve çentik bir anda karşı kenarda beliriyorsa, tek başına güzel çizilmiş iki kare arasında hareketin açıklayamadığı bir değişiklik yakalamışız demektir.

Bölüm 13 · Videoda ses ve görüntü uyumu, olay sırası ve nedensellik

Tokmak iki kez vuruldu. Kapıyı usta mı açtı?

Dört saniyelik kaydın başında yeşil kapı kapalı. Dışarıdan uzanan bir el pirinç halkayı kaldırıp bırakıyor. Halka metal levhaya vuruyor. Tak. Kapı hâlâ kapalı. Aynı el halkayı bir daha kaldırıp bırakıyor. İkinci bir tak sesi duyuluyor. El geri çekiliyor. Kısa bir bekleyişten sonra kapı içeri doğru açılıyor. Kamera yalnız kapının üst yarısını çekmiş. Kapıyı açanın eli de yüzü de görünmüyor.

Çırak bu kaydı bakım defterine geçirmek istiyor. Telefonu ustaya getiriyor. “Sana önce sesini dinleteyim,” diyor. “Sonra görüntüsüne de bakıp ne yazacağımıza karar verelim.” Usta kalemini çıkarıyor. Çırak kaydı başa sarıp ekranı aşağı çeviriyor.

Önce yalnız dinleyelim

Tak. Kısa bir ara. Tak. İkinci sesin ardından ince bir çınlama kalıyor. Sonra kayıt bitiyor.

Siz deftere ne yazardınız? Usta “İki vuruş duydum,” diye başlıyor. Çırak bekliyor. “Bizim pirinç tokmağa benziyor,” diye ekliyor usta. İki cümleyi nasıl sınayabiliriz? İlkini kontrol etmek için kaydı başa sarıp vuruşları birlikte sayabiliriz. İkincisini sınamanın bir yolu, onuncu bölümde yaptığımız gibi tokmakla tezgâhtan aldığımız örnekleri dinleyip çınlamaları karşılaştırmak. Yakında başka bir metal parçaya vurulsa benzer bir ses çıkabilir. Ustanın “benziyor” derken pay bıraktığı yer burası.

Şimdi sesi kapatıp ekranı çevirelim. Bir el alttan uzanıyor, halkayı kaldırıp bırakıyor. Halka levhaya dönüyor. El hareketi tekrarlayıp görüntüden çekiliyor. Biraz sonra kapıyla kasa arasında ince bir aralık beliriyor ve genişliyor. Videoyu durdurduğumuzda içeride kimin bulunduğunu göremiyoruz.

“İki defa kaldırdı,” diyor çırak. Bu sefer parmağıyla gösterebileceği iki an var. Ama bu sessiz görüntüyü tek başına izleseydik tokmağın ikinci vuruşta uzun uzun çınladığını söyleyebilir miydik? Halkanın

levhaya kadar geldiğini görmek, çıkan sesi hayal etmemizi kolaylaştırıyor. Çınlamanın süresini ölçmek için yine ses kaydına ihtiyacımız var.

Sesle görüntüyü bir araya getirince ne öğreniyoruz?

Çırak sesi açıyor. İlk “tak”, halka levhaya döndüğü sırada duyuluyor. İkincisinde de öyle. Kapı ikinci vuruştan sonra aralanıyor. Şimdi elimizde iki ayrı “iki” sayısından daha fazlası var. **Hangi sesin hangi hareketle aynı zamana geldiğini** söyleyebiliyoruz.

Bunun önemini küçük bir kurgu deneyiyle görebiliriz. Çırak kaydın bir kopyasında yalnızca sesi bir saniye ileri kaydırsın. Görüntüdeki el yine iki kez halkayı kaldırır. Sesi tek başına dinlediğimizde hâlâ iki vuruş sayarız. Oysa beraber oynatınca halka levhaya değdiğinde ortalık sessiz kalabilir, el çekildikten sonra da bir “tak” duyabiliriz. Her iki parçayı ayrı ayrı doğru özetleyen bir sistem, yalnız o özetleri yan yana koyarsa bu farkı kaçırabilir.

Birden fazla türden bilgiyi birlikte kullanan modellere **çok kipli** (*multimodal*) deniyor. Bizim örneğimizde böyle bir modelin izleyebileceği yollardan birini düşünelim. Görüntüyü işleyen bölüm, art arda gelen karelerde halkanın levhaya yaklaşmasını bir grup sayıyla temsil ediyor. Sesi işleyen bölüm de kısa bir zaman dilimindeki darbenin ani başlangıcını ve ardından kalan titreşimi sayılarla ifade ediyor. Onuncu bölümde gördüğümüz gibi tek bir görüntü karesinin süresine çok sayıda ses ölçümü sığabilir. İki taraftaki bilgiyi ilişkilendirmek için bunların kaydın *hangi zaman aralığından* geldiğini korumak işe yarar.

Peki modelin o ilişkiye dikkat etmesini ne sağlayabilir? Bir eğitim alıştırması düşünelim. Çırak bazı örneklerde sesi yerinde bırakıyor, bazılarında az önceki gibi kaydırıyor. Modelden her seferinde sesle hareketin birbirine uyup uymadığını tahmin etmesini istiyor. İki örnekte de aynı kapı, aynı el ve aynı çınlama var. Yalnız kapıyı tanımakla ya da yalnız sesi saymakla ikisini ayırmak güç. Darbenin başlangıcını halkanın dönüşüyle birlikte değerlendiren bir tahmin ise daha başarılı olabilir. Çırak doğru cevabı bildiği için hatayı ölçebilir. İlk bölümde tanıştığımız öğrenme fikri burada da geçerli. Bu kez ağırlıkları ayarlamaya yarayan hata, bir siparişin süresinden değil, iki olayın zaman bakımından uyuşup uyuşmadığına verilen cevaptan geliyor.

Model kapının açılmasını nasıl öğrenir, aynı hareketi nasıl üretir?

Usta ekrandaki kapı aralığını gösteriyor. “Sesi tamamen kapatsak bir model yine de kapının açıldığını öğrenebilir mi?” Çırak iki kısa film hazırlıyor. İlkinde kaydın yalnızca kapının hareket ettiği son kısmı var. İkincisinde o kısmı tersten oynatıyor. İki filmde de aynı yeşil boya, aynı tokmak, aynı ışık görünüyor. Birinde kapı açılıyor, ötekinde kapanıyor.

Bu kez eğitimde her filmin yanına hareketin adını yazalım. Model iki filme de “açılıyor” cevabını verirse birinde yanılacak. Kasayla kapının kenarını peş peşe birkaç karede ilişkilendiren bir hesap düşünün. Aradaki ışık şeridi bir filmde inceden kalına, ötekinde kalından inceye gidiyor. Bu farkı hesaba katmak tahmin hatasını azaltabilir. Eğitimde ağırlıklar küçük adımlarla ayarlandıkça model böyle ipuçlarından yararlanmayı öğrenebilir. Bunu sınamak için çırak başka bir kapıyı farklı bir açıdan çeker. Yeni çekimde ışık şeridi görünmese bile kanadın kasaya göre nasıl döndüğü hâlâ seçilebilir. Model burada zorlanırsa ilk örneklerdeki başarısının hangi koşullara bağlı olduğunu araştıracak bir nedeni vardır.

Çırak şimdi ustaya kapının kapalı olduğu ilk kareyi gösteriyor. “Peki bu resmi verip ‘Tokmak iki kez vurulsun, sonra kapı açılsın’ dersem?” Bu isteğin cevabı dört saniye sürecek. Halka birinci darbeyle ikinci darbe arasında yeniden kalkmalı. İki darbe boyunca kapı kasaya dayalı kalmalı. Kapı ancak ondan sonra hareket etmeli. Tek tek bütün resimler güzel çizilmiş olsa bile kapı ilk darbede açılırsa film tarife uymaz.

On ikinci bölümde gördüğümüz üretim yöntemini bu örneğe uygulayalım. Eğitim sırasında yazılı tarifi bulunan bir filmin karelerini temsil eden sayılara program rastgele gürültü ekliyor. Eklediği değerleri bildiği için modelin gürültü tahminindeki hatayı ölçebiliyor. Diyelim ki bir karede halkanın alt kenarı gürültü içinde güçlükle seçiliyor. Önceki ve sonraki karelerde halka yukarı kalkıyorsa aradaki kare için makul konumla, levhaya doğru iniyorsa makul konum farklıdır. Model, komşu karelerden ve tariften aldığı bilgiyi tahmininde kullanarak bu hatayı azaltabilir.

Yeni filmi üretirken çırak ilk fotoğrafı ve tarifi veriyor. Geri kalan anlar için sistem rastgele sayılardan başlayıp kısa görüntü dizisini adım adım belirginleştiriyor. Usta iki deneme istiyor. Bir filmde ikinci vuruştan sonra kapı hemen açılıyor. Ötekinde halka biraz sallanıyor, sonra kapı açılıyor. İkisi de tarife uyabilir. Çırak ikinci filmi durduruyor. “Şu aralıktan ustanın önlüğü görünseydi, deftere kapıyı ustanın açtığını yazar mıydık?” Üretilmiş film için bu, anlatıma

uyan bir ayrıntı olabilir. Bakım defterine ne yazacağımıza karar vermek içinse telefondaki asıl kaydı yeniden açmamız gerekir.

“Sonra” ile “çünkü” arasında ne var?

Çırak asıl kaydı ses ve görüntüyü birlikte işleyebilen bir modele verdiğinde şöyle bir cevap aldığını düşünelim.

> Tokmak iki kez vurulunca usta sesi duyup kapıyı açıyor.

Usta kalemini defterin üstünde bekletiyor. “Benim duyduğumu hangi saniyede anladın?”

Cümleyi baştan izleyelim. Tokmağın iki kez vurulduğu bölüm için hem halka hareketini gösterebiliyor hem de darbeleri dinletebiliyoruz. Kapının daha sonra açıldığını görmek için son karelere gidebiliriz. “Usta sesi duydu” kısmına geldiğimizde ise durdurabileceğimiz bir kare bulamıyoruz. İçeriden bir cevap da duyulmuyor. Kapıyı açan kişiyi hiç görmedik. Dışarıdaki elin sahibi hakkında da bu kadar dar bir çekimden pek az şey söyleyebiliyoruz.

Oysa modelin cümlesi neden kulağa doğal geliyor? Bir atölyenin kapısı çalınınca içerideki ustanın açması tanıdık bir hikâye. Hatta modelin tahmini bu kez doğru çıkabilir. Küçük bir sınama yapalım. İçeride başka birinin tam o sırada kapıya doğru yürüdüğünü ve çıkmak için kapıyı açtığını düşünün. Dışarıdaki telefonun bu dört saniyede kaydettiği seslerden veya görüntülerden hangisi mutlaka değişirdi? İki hikâye de aynı kayda uyuyorsa birini seçmek için ek bilgi gerekir. İçeriyi gören ikinci bir kamera kapıyı kimin açtığını gösterebilir. O kişinin vuruşu duyup duymadığını öğrenmek içinse kendisine sormak daha yerinde olur.

Çırak sonunda deftere şu cümleyi geçiriyor.

> Tokmak iki kez vuruldu. İkinci vuruştan kısa süre sonra kapı aralandı. Kapıyı açan kişi görüntüye girmedi.

Usta defteri ilk sayfasına çeviriyor. On parça 25 dakika, yirmi parça 45 dakika. Çırak o gün aradaki hesabı yapıp sipariş başına beş dakikalık bir hazırlık süresi düşünmüştü. “Hazırlık gerçekten beş dakika mı sürüyordu?” diye soruyor usta. Çırak eski kayıtlara bir daha bakıyor. Defterde yalnızca toplam süreler yazılı.

O sırada kapıya yeni bir sipariş geliyor. Usta makinenin başına geçerken çırak saatine bakıyor. Bu kez deftere iki ayrı saati yazmak için yer açıyor. Birini usta hazırlığa başlarken, ötekini ilk parçayı işlemeye başladığında dolduracak.

Genel Değerlendirme ve Kapanış

Ustanın defterindeki iki siparişten yola çıkıp hazırlığın beş dakika sürdüğünü hesaplamıştık. Hesap her iki siparişe de uyuyordu. Kitabın sonunda aynı sayfaya döndüğümüzde bu kez orada eksik olanı gördük. Usta, hazırlığın ne zaman bittiğini hiç kaydetmemişti. Aradaki beş dakikayı hazırlığa bağlamak bizim varsayımımızdı. Bir modelin nasıl öğrendiğini anlamak da biraz böyle bir ayrımı fark etmekle başlıyor.

Eğitim sırasında model bir tahmin yapar. Bu tahmini değerlendirecek bir ölçü seçer, modelin ayarlanabilir sayılarını bu ölçüye göre değiştiririz. Sipariş örneğinde tahminle ölçülen süre arasındaki farkı kullandık. Dil modelinde ise yarım bırakılan cümlenin metinde nasıl devam ettiğini biliyorduk. Model bu devama ne kadar az olasılık verdiyse kaybı o kadar büyük oldu. Her iki örnekte de nasıl bir ilişki öğrenilebileceğini, modele gösterdiğimiz örnekler kadar ona sorduğumuz soru da belirledi.

Eğitilmiş modeli kullanırken başka bir değişikliği izledik. Ustanın nereye gideceğini anlatan cümleyi çıkarınca “Kapıyı neden kilitledi?” sorusuna verebileceğimiz cevap değişti. Bunun için modelin yeniden eğitilmesi gerekmedi. Değişen, cevap verirken elinde bulunan bilgiydi. Atölyenin kapısıyla ilgili iki soru bu ayrımı daha da belirginleştirdi. “Tokmak neden söküldü?” sorusunu araştırırken bakım defterindeki kaydın doğru kapıya ait olup olmadığına baktık. “Kapı şu anda kilitli mi?” sorusunda ise doğru cihazı bulduktan sonra bile ölçümün saatini bilmemiz gerekti.

Görüntü, ses ve video bölümlerinde bir modelin öğrendiklerini kullanarak nasıl yeni içerik üretebildiğini gördük. “Yeşil kapıya pirinç bir tokmak tak” tarifi, görünüşleri birbirinden farklı birçok resme uyabilir. Ustanın tokmağındaki çentik önemliyse fotoğrafı açıp tam o noktayı karşılaştırabiliriz. Ses için de benzer bir deney yapabiliriz. İki kişinin söylediği “Kapı açık” sözü yazıya aynı biçimde geçer. Bu yazıyı yeniden seslendirdiğimizde kelimeler korunabilir, ama konuşmacının ilk kayıtta nerede durakladığını öğrenmek için kaydı dinlememiz gerekir. Üretim sırasında sistem, tarifte açık bırakılan ayrıntılara da bir biçim verir.

Kendi işinizden bir örnek seçtiğinizde küçük bir deneme yapabilirsiniz. Önce girdide tek bir şeyi değiştirin ve bunun cevabı nasıl etkilemesini

beklediğinizi yazın. Örneğin kapı videosunu geriye doğru oynattığınızda açılma hareketi kapanmaya dönüşür. Kapının rengi ise aynı kalır. Ardından modelin cevabına bakın. Bu karşılaştırma, “Videoyu anladı mı?” gibi geniş bir soruyu, sonucunu kendi gözünüzle değerlendirebileceğiniz iki ayrı soruya dönüştürür.

Çırağın son sayfada yaptığı da buna benzerdi. Bir sonraki sipariş için daha ayrıntılı bir açıklama yazabilirdi. O, eski hesabında en çok merak ettiği şeyi ölçmeye karar verdi. Bundan sonra bir modelin cevabında sizi düşündüren bir ayrıntıyla karşılaştığınızda kendinize aynı soruyu sorabilirsiniz. *Bunu öğrenmek için defterime neyi kaydetmeliydim?*

Sözlük

Bu sözlükte her kavramın İngilizce karşılığını ve kitapta ele alındığı bölüm numarasını bulabilirsiniz. Bir tanımı atölyedeki örneğiyle birlikte hatırlamak için belirtilen bölüme dönebilirsiniz.

A

Ağırlık (*weight*, Bölüm 1) — Modelin bir girdinin hesaba katkısını ayarlayan öğrenilebilir sayısı. Eğitim sırasında tahmin hatasına göre güncellenebilir.

Aktivasyon fonksiyonu (*activation function*, Bölüm 1) — Bir katmanda hesaplanan değeri sonraki işleme aktarırken dönüştüren kural. ReLU, negatif değerleri sıfıra çeker.

Araç çağrısı (*tool call*, Bölüm 5) — Modelin, uygulamanın tanıdığı bir işlemi belirli girdilerle çalıştırması için ürettiği istek. Kilidi okuyan uygulama sonucu modele geri verir.

B

Bağlam (*context*, Bölüm 3–6) — Bir cevabı üretirken modelin yararlanabildiği girdiler. Soru, önceki sözler ve getirilen kayıtlar buna dahil olabilir.

Ç

Çok kipli (*multimodal*, Bölüm 13) — Metin, ses ve görüntü gibi birden fazla bilgi türüyle çalışabilen sistem.

D

Difüzyon (*diffusion*, Bölüm 8 ve 12) — Bu kitaptaki örnekte, gürültülü bir temsile adım adım düzeltmeler uygulayarak görüntü ya da video üretme yöntemi.

Dikkat (*attention*, Bölüm 3 ve 7) — Başka konumlardan gelen bilgilerin mevcut konumdaki hesaba hangi paylarla katılacağını belirleyen mekanizma.

E

Eğitim (*training*, Bölüm 1–3) — Örneklerden hesaplanan hata yardımıyla modelin öğrenilebilir parametrelerini ayarlama süreci.

Etiket (*label*, Bölüm 10) — Eğitim örneğine verilen hedef cevap. Ses kaydı için “tokmak” bir etiket olabilir.

F

Frekans (*frequency*, Bölüm 10) — Bir titreşimin bir saniyede tamamladığı çevrim sayısı. Birimi hertzdir.

G

Geri yayılım (*backpropagation*, Bölüm 1) — Sonuçta hesaplanan kaybın önceki katmanlardaki parametrelere duyarlılığını zincir kuralıyla hesaplama yöntemi.

Gizil temsil (*latent representation*, Bölüm 8) — Görüntü gibi bir girdinin özelliklerini kodlayan ara sayısal gösterim.

Gömme (*embedding*, Bölüm 3 ve 6) — Bir token ya da metin parçası gibi bir öğenin öğrenilmiş sayısal temsili.

K

Kare (*frame*, Bölüm 12) — Videonun belirli bir anına karşılık gelen görüntü. Karelerin sırası hareketin yönü hakkında bilgi taşır.

Kayıp (*loss*, Bölüm 1–2) — Tahminle hedef arasındaki uyuşmazlığı eğitim için sayıya dönüştüren ölçü.

M

Maske (*mask*, Bölüm 9) — Görüntüde hangi konumların düzenlemeye katılacağını belirleyen seçim.

Ö

Örnekleme hızı (*sampling rate*, Bölüm 10) — Ses kaydında bir saniyede alınan ölçüm sayısı.

P

Parametre (*parameter*, Bölüm 1) — Modelin eğitimle ayarlanabilen sayısı. Ağırlıklar bu sayıların bir örneğidir.

Piksel (*pixel*, Bölüm 7–9) — Sayısal görüntünün bir konumunda renk ya da parlaklık taşıyan öğe.

Prozodi (*prosody*, Bölüm 11) — Konuşmadaki vurgu, süre, ritim ve ezgi özellikleri.

R

RAG (*retrieval-augmented generation*, Bölüm 6) — İlgili belgeleri veya parçalarını bulup cevap üreten modele bağlam olarak verme yaklaşımı.

S

Softmax (*softmax*, Bölüm 3) — Adayların ham puanlarını, toplamı bir olan pozitif paylara dönüştüren işlem.

Spektrogram (*spectrogram*, Bölüm 10–11) — Hangi frekansların hangi zaman aralığında ne kadar güçlü olduğunu gösteren tablo.

T

Tohum (*seed*, Bölüm 8) — Sözde rastgele sayı üretecinin başlangıcını belirleyen değer.

Token (*token*, Bölüm 2–4) — Dil modelinin işlediği metin birimi. Bir kelimeyi, kelime parçasını veya noktalama işaretini temsil edebilir.

V

Vektör (*vector*, Bölüm 3 ve 7) — Sırası belirli sayılardan oluşan liste. Bir token'ın özellikleri veya bir görüntü parçası bu biçimde temsil edilebilir.

Vocoder (*vocoder*, Bölüm 11) — Kitaptaki konuşma sentezi örneğinde, akustik temsilden hoparlörde çalınabilecek ses dalgası üreten bileşen.