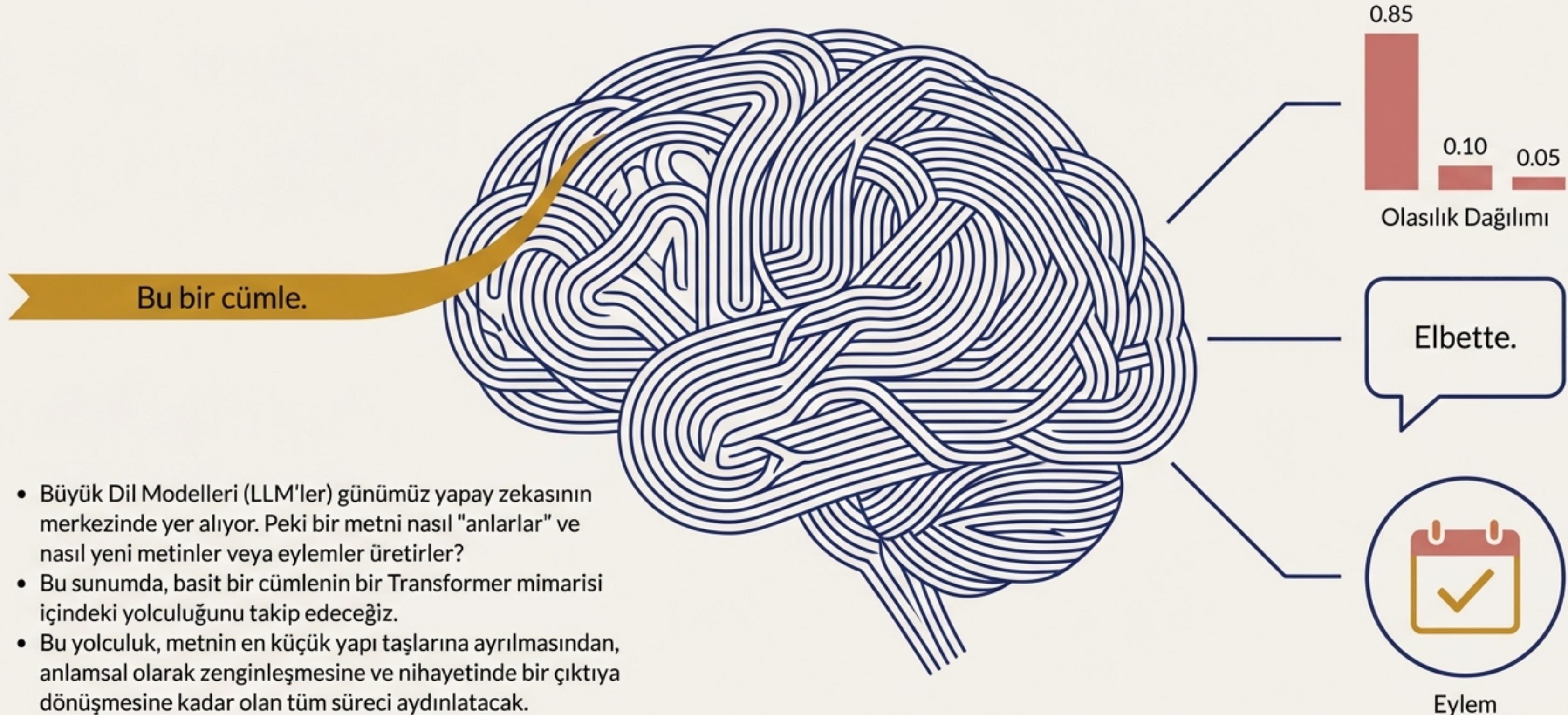


Büyük Dil Modellerinin Anatomisi: Bir Cümplenin Veri İçindeki Yolculuğu

Bir metnin, ham bilgiden eyleme dönüşümünü adım adım keşfedin.



Yol Haritamız: Bir Fikrin Olasılığa Dönüşümü



Bu sunumda, bir cümlenin bu beş temel istasyondan nasıl geçtiğini ve her birinde nasıl bir dönüşüm yaşadığını inceleyeceğiz. Her adım, bir öncekinin üzerine inşa edilerek modelin karmaşık anlama ve üretme yeteneğini ortaya çıkarır.

Adım 1: Metni Anlamlı Yapı Taşlarına Ayırmak (Tokenizasyon)

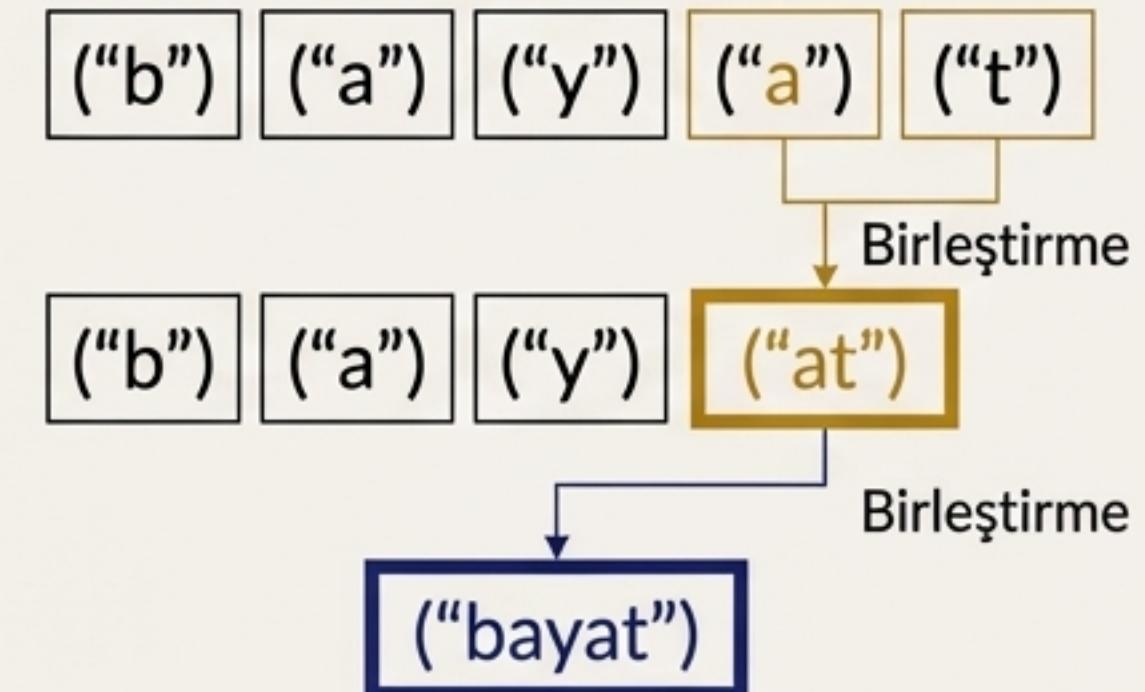
Bir modelin metni işleyebilmesi için onu önce yönetilebilir parçalara ayırması gereklidir. Bu parçalara 'token' denir.

- **Problem:** Sadece kelime lere ayırmak yeterli değildir. Model, eğitim setinde görmemiş olduğu kelimelerle ('out-of-vocabulary') veya 'koşuyordum', 'koşacak' gibi eklerle nasıl başa çıkar?
- **Çözüm:** Subword (Alt Kelime) Tokenizasyonu. Kelimeler, daha küçük ve anlamlı birimlere ayrılır. Bu, modelin dilin morfolojik yapısını öğrenmesini ve yeni kelimeler oluşturabilmesini sağlar.
- **Popüler Bir Yöntem:** BPE (Byte-Pair Encoding). Veri odaklı bir yaklaşımdır. Metinlerde en sık tekrar eden karakter çiftlerini yinelemeli olarak birleştirerek bir 'token' sözlüğü oluşturur.
 - **Avantajları:**
 - **Genelleme:** Bilinmeyen kelimeleri bilinen alt kelimelerden oluşturur.
 - **Verimlilik:** Sözlük boyutunu kontrol altında tutar.
 - **Esneklik:** Dilden bağımsız çalışabilir.

Subword Örneği tokenizasyon



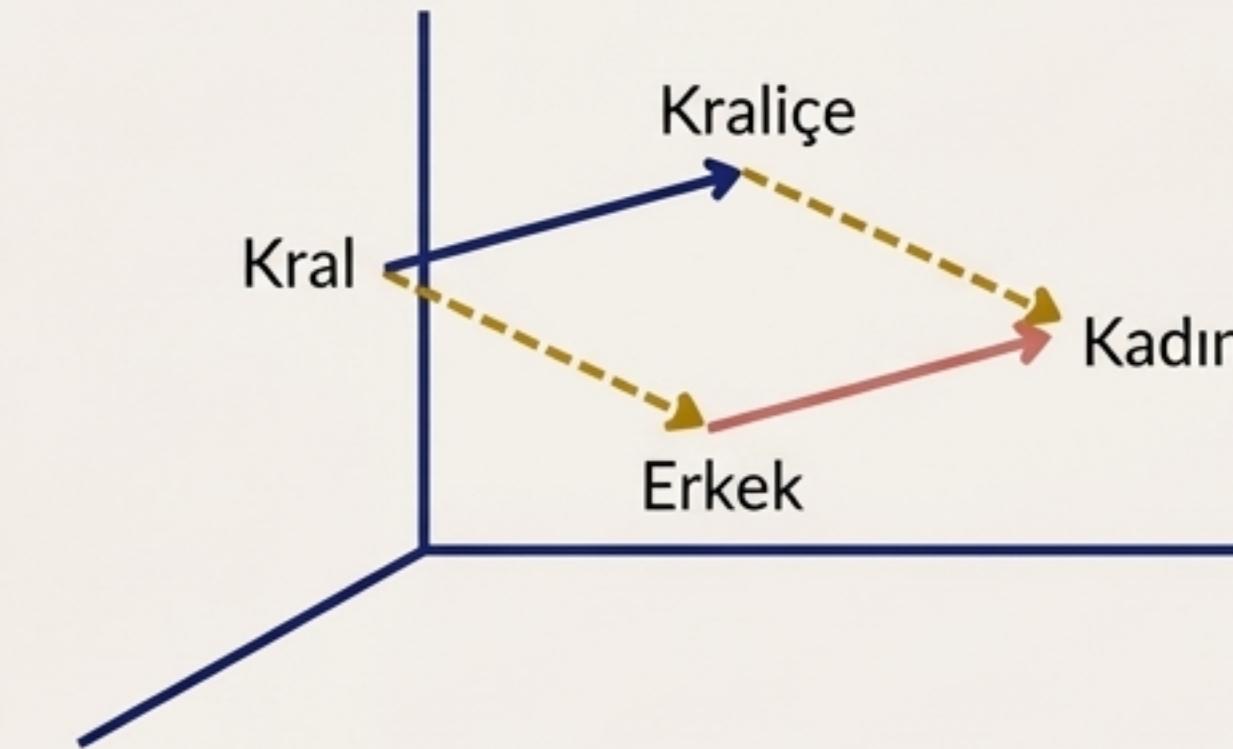
BPE Algoritması



Adım 2: Kelimeleri Vektörlere Dönüşütmek (Embedding)

Token'lar, modelin anlayabileceği sayısal temsillere, yani yüksek boyutlu vektörlere dönüştürülür. Bu işlem "embedding" denir.

- Her token için, öğrenilmiş bir "Embedding Matrisi" ('We') üzerinden karşılık gelen vektör alınır.
- Bu vektörler, kelimenin anlamsal özünü kodlar.
- **Asıl Güç:** Bu yüksek boyutlu uzayda, yönler ve mesafeler anlamsal ilişkileri temsil eder. **Benzer anlama gelen kelimeler** birbirine yakınlıkta, belirli yönler "cinsiyet", "çoğulluk" veya "ülke-başkent" gibi ilişkileri kodlayabilir.



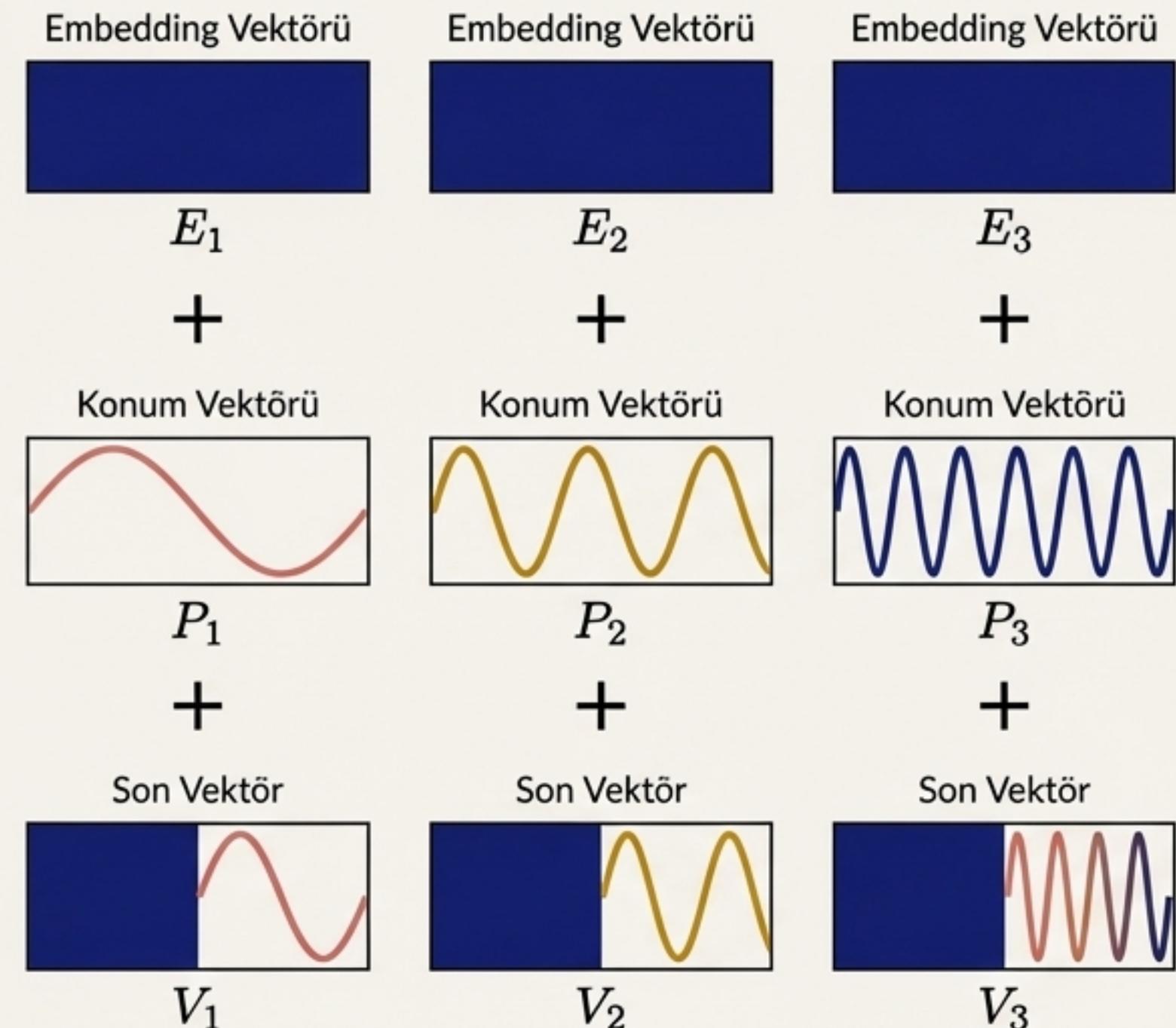
$$\text{Vektör(Kral)} - \text{Vektör(Erkek)} + \text{Vektör(Kadın)} \approx \text{Vektör(Kralice)}$$

Adım 3: Sıranın Önemini Korumak (Positional Encoding)

Transformer mimarisi, RNN'lerin aksine kelimeleri sıralı olarak değil, aynı anda işler. Bu, muazzam bir paralelleştirme ve hız avantajı sağlar.

- **Problem:** Bu paralel işleme sırasında kelimelerin orijinal sırası kaybolur. "Kedi köpeği kovaladı" ile "Köpek kediyi kovaladı" cümleleri ayırt edilemez hale gelir.
- **Çözüm:** Positional Encoding (Konumsal Kodlama). Her kelimenin embedding vektörüne, o kelimenin cümledeki konumunu belirten benzersiz bir vektör eklenir.

Bu vektörler genellikle farklı frekanslardaki sinüs ve kosinüs fonksiyonları ile oluşturulur. Bu sayede model, kelimelerin mutlak ve göreceli konumlarını hakkında bilgi sahibi olur.

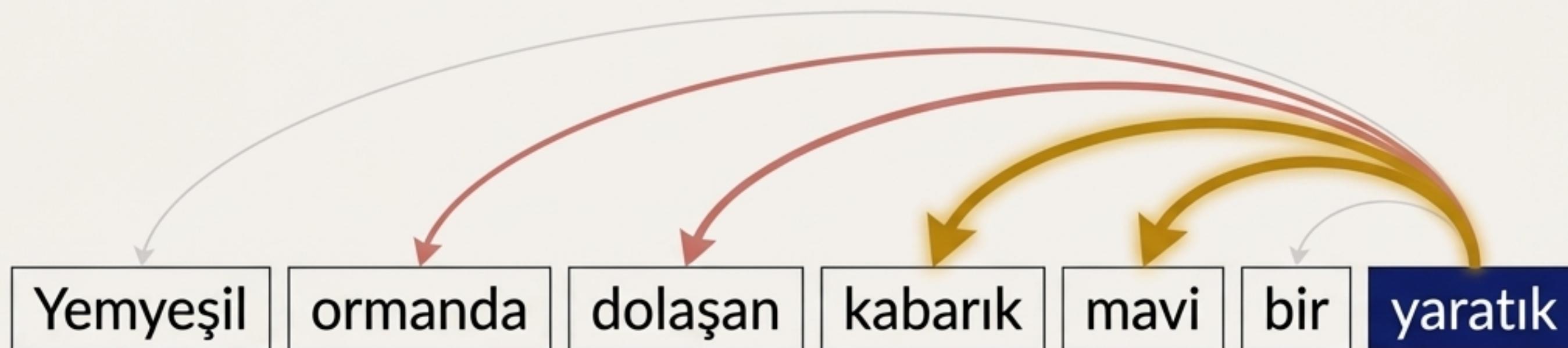


$$\text{Son Vektör} = \text{Embedding Vektörü} + \text{Konum Vektörü}$$

Transformer'ın Kalbi: Self-Attention Mekanizması

Modelin bağlamı anlama yeteneği '**Self-Attention**' (**Öz-Dikkat**) mekanizmasından gelir. Bu mekanizma, her kelimenin cümledeki diğer tüm kelimelerle olan ilişkisini ve önemini dinamik olarak hesaplamasını sağlar.

- **Neden Gerekli?** Bir kelimenin anlamı bağlama göre değişir.
 - Örnek 1: 'Yüzünde bir **ben** vardı.' (İsim) vs. 'Bu işi **ben** yaptım.' (Zamir)
 - Örnek 2: '**Eyfel**' kelimesini gören model, '**Kulesi**' kelimesine daha fazla dikkat etmesi gerektiğini anlar.
- **İşleyiş:** Her kelime, kendi anlamını zenginleştirmek için diğer kelimelere 'bakar' ve hangilerinin kendisiyle en ilgili olduğuna karar verir. Bu 'ilgi' skorlarına göre diğer kelimelerden bilgi toplar. Bu süreç, kelime vektörlerini bağlama duyarlı hale getirir.



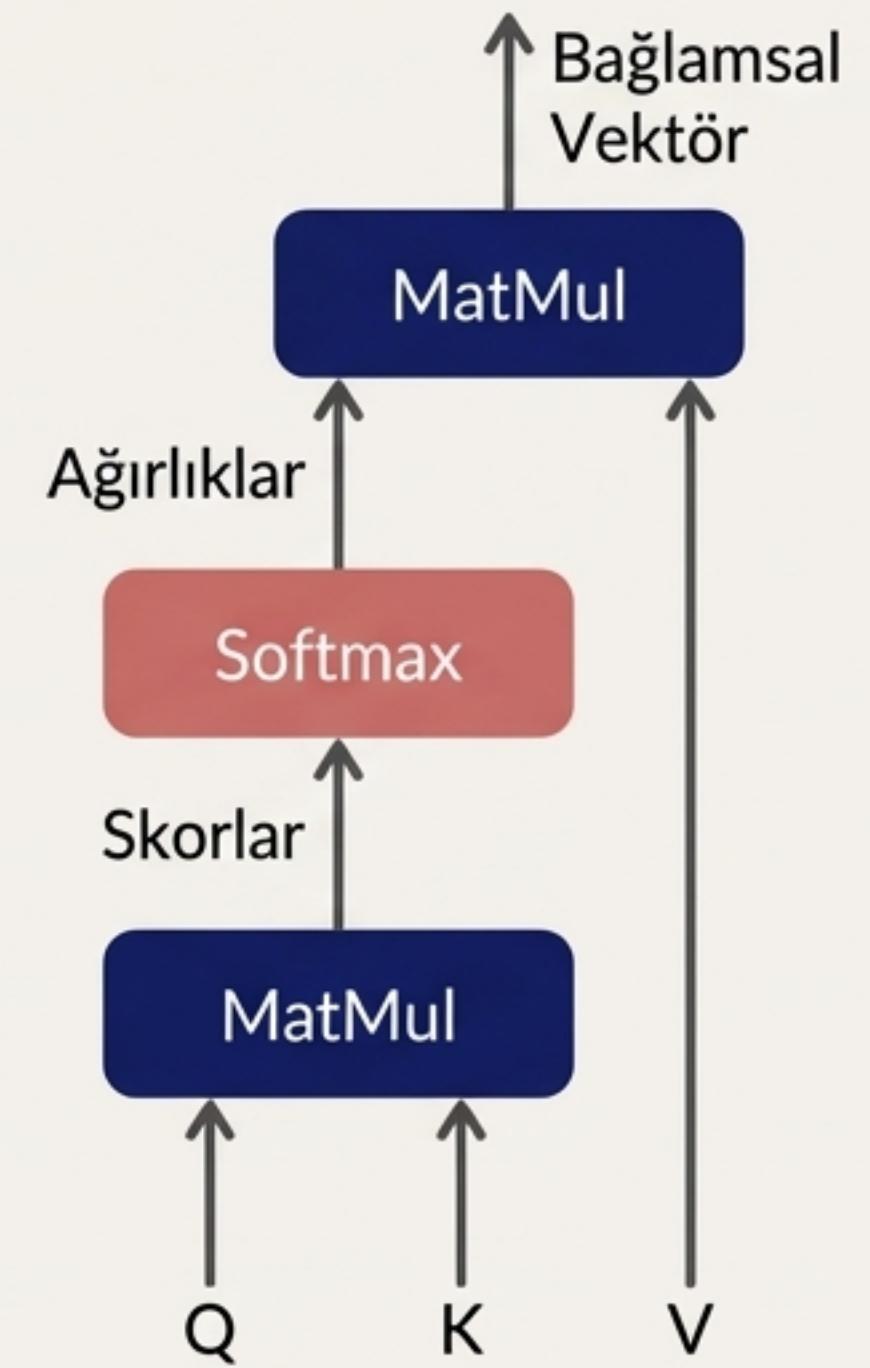
Attention'ın Arkasındaki Sezgi: Sorgu, Anahtar ve Değer

Self-Attention, her kelime vektöründen üç farklı vektör üreterek çalışır:

- **Query (Q - Sorgu):** O anki kelimenin 'Ben ne arıyorum?' sorusunu temsil eder.
- **Key (K - Anahtar):** Diğer kelimelerin 'Ben hangi bilgiyi taşıyorum?' etiketidir.
- **Value (V - Değer):** Diğer kelimelerin 'Eğer benimle ilgiliysem, sana vereceğim asıl bilgi budur' içeriğidir.

Süreç:

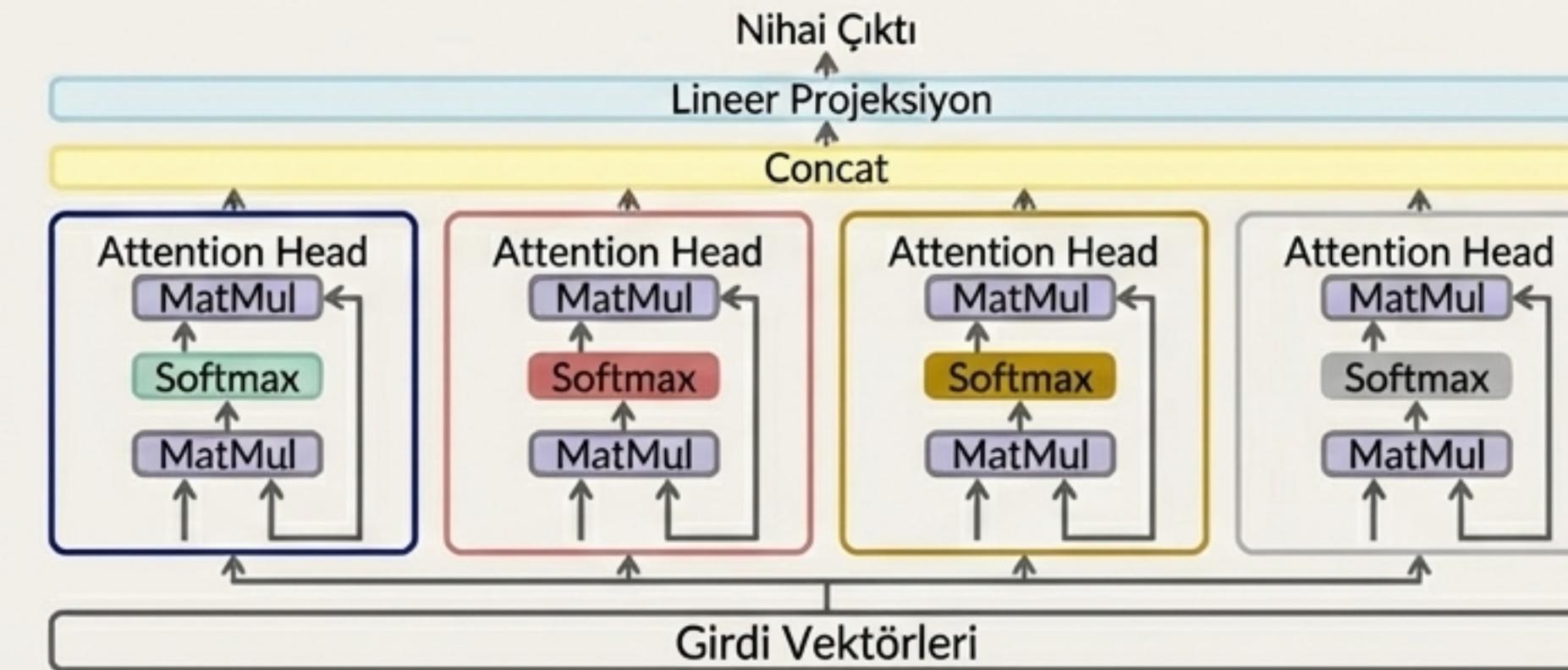
1. **Skorlama:** Bir kelimenin **Query'si**, diğer tüm kelimelerin **Key'leri** ile karşılaştırılır (genellikle dot-product ile). Bu, 'ilgi skorlarını' oluşturur.
2. **Normalleştirme (Softmax):** Skorlar, toplamları 1 olan ağırlıklara dönüştürülür. Bu, dikkatin nereye dağılacagını belirler.
3. **Ağırlıklı Toplama:** Her kelimenin **Value'su**, hesaplanan ağırlıkla çarpılır ve hepsi toplanır. Sonuç, o kelime için bağlamdan zenginleşmiş yeni vektördür.



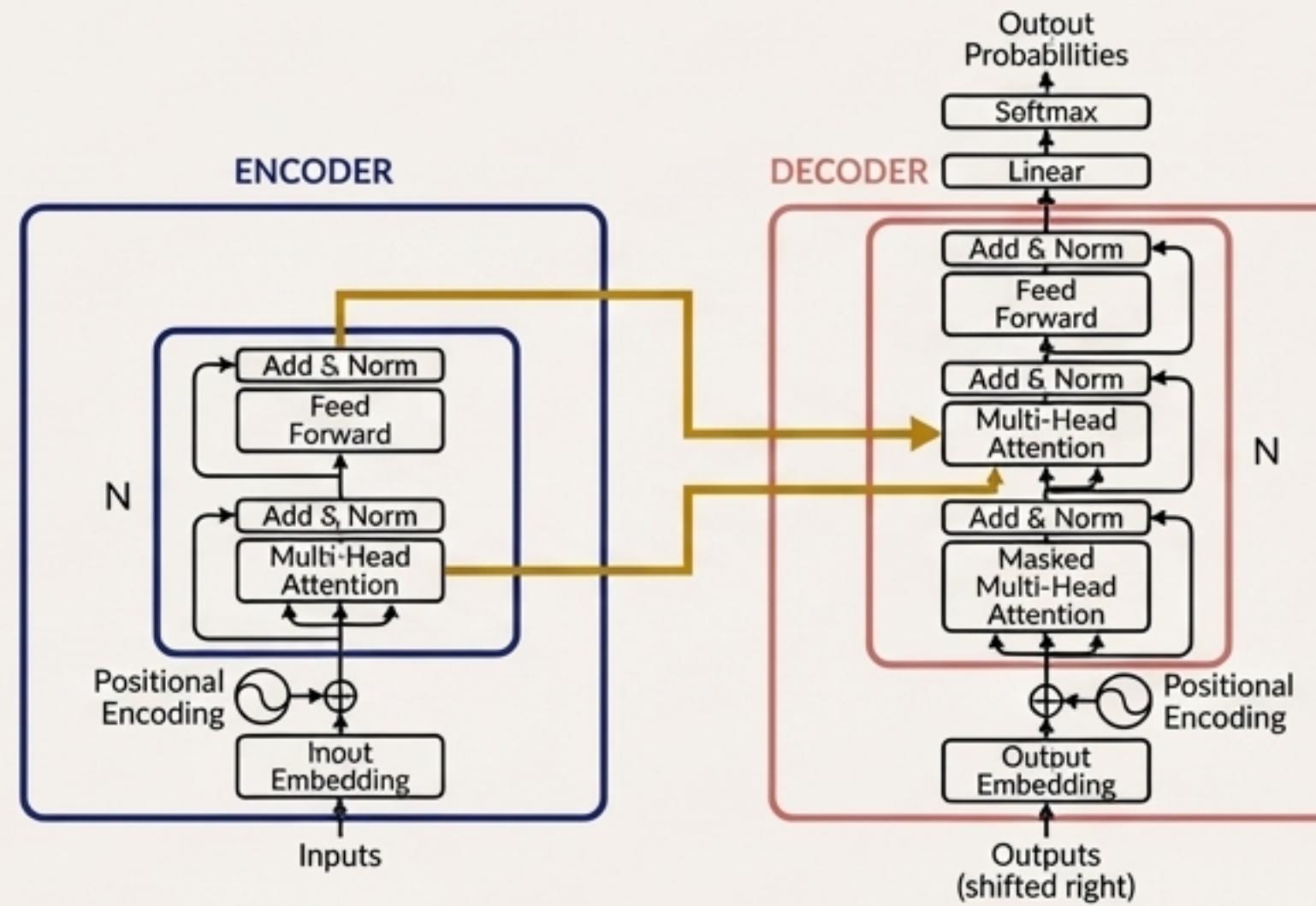
Farklı Perspektifler Kazanmak: Multi-Head Attention

Bir cümlenin içinde aynı anda birden çok türde ilişki bulunur (dilbilgisel, anlamsal, zamir referansları vb.). Tek bir attention mekanizması tüm bu ilişkileri yakalamakta zorlanabilir.

- **Çözüm: Multi-Head Attention (Çok Başlı Dikkat).** QKV süreci, farklı ve öğrenilmiş projeksiyon matrisleri kullanılarak paralel olarak birden çok kez ("h" defa, örn: 8 veya 96 kafa) çalıştırılır.
- Her 'kafa', farklı bir temsil alt uzayındaki ilişkilere odaklanmayı öğrenir.
- Bir kafa, "o" zamirinin kime işaret ettiğine odaklanabilir.
- Başka bir kafa, sıfatların hangi isimleri nitelediğine odaklanabilir.
- Bir diğeri, cümlenin genel anlamsal yapısına odaklanabilir.
- Tüm kafaların çıktıları birleştirilir ve nihai, çok yönlü ve zenginleştirilmiş bir vektör oluşturulur.



Mimarinin Yapı Taşları ve Farklı 'Yolculuklar'



Encoder (Sol Taraf): Giridi metnini okur ve bağlamsal olarak zengin bir dizi sayısal temsil (Key ve Value vektörleri) oluşturur.

Görevi: Metni derinlemesine anlamak.

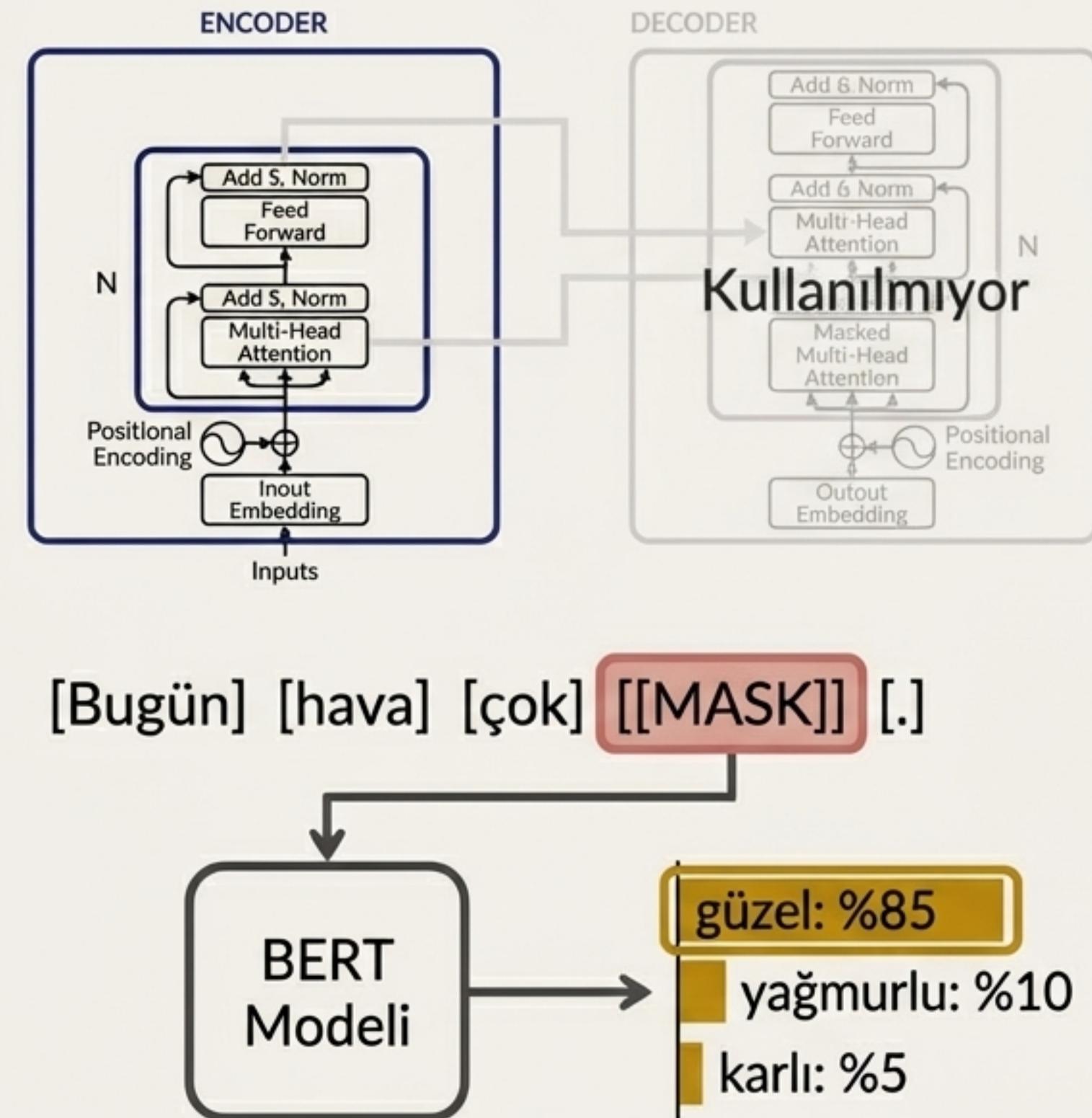
Decoder (Sağ Taraf): Encoder'dan gelen temsilleri ve o ana kadar ürettiği kelimeleri kullanarak bir sonraki kelimeyi tahmin eder.

Görevi: Yeni metin üretmek.

Bu temel yapı, modern LLM'lerin üç ana ailesini doğurmuştur.

Anlama Odaklı Modeller: Encoder-Only (Örn: BERT)

- **Anlatı:** Yolculüğün sadece ilk yarısı.
- **Mimari:** Sadece Transformer'ın Encoder bloğunu kullanır.
- **Temel Yetenek:** Metni çift yönlü (bidirectional) olarak anlama. Bir kelimenin anlamını hem öncesindeki hem de sonrasındaki kelimelere bakarak çıkarır.
- **Ön-Eğitim Görevi:** Masked Language Modeling (MLM). Cümledeki bazı kelimeler rastgele [MASK] token'ı ile gizlenir ve modelden bu gizlenmiş kelimeleri tahmin etmesi istenir. Bu, modelin derin bağlamsal anlayış geliştirmesini zorlar.
- **Kullanım Alanları:** Metin sınıflandırma, duyu analizi, varlık tanıma (NER), soru-cevap sistemleri (anlama kısmı). Metin üretmek için tasarlanmamıştır.



Üretme Odaklı Modeller: Decoder-Only (Örn: GPT)

Anlatı: Yolculuğun sadece ikinci yarısı.

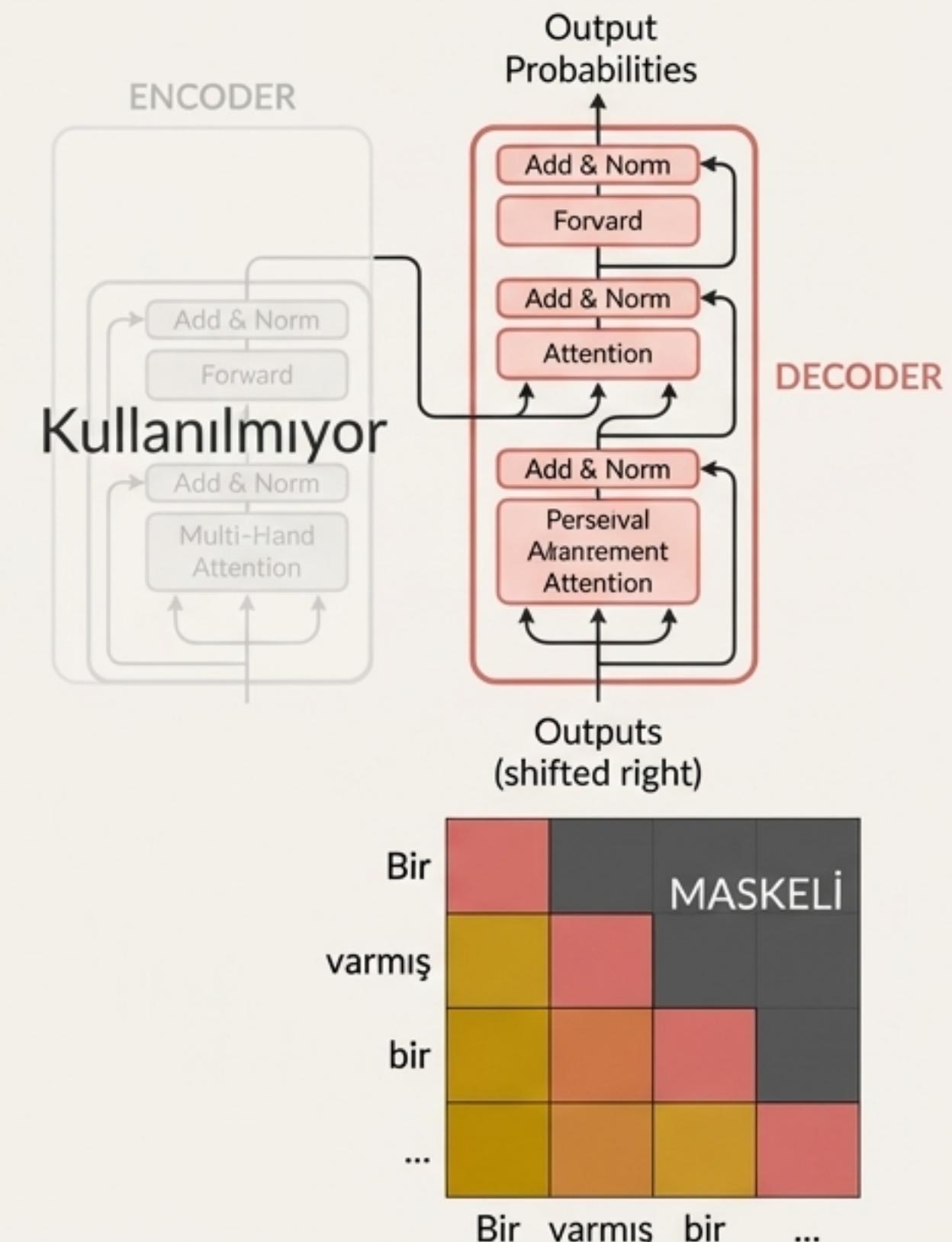
Mimari: Sadece Transformer'ın Decoder bloğunu kullanır.

Temel Yetenek: Metni otoregresif (autoregressive) olarak, yani bir sonraki kelimeyi o ana kadar üretilmiş kelimelere dayanarak üretme.

Kritik Özellik: Causal (Nedensel) Maskeleme. Self-Attention mekanizması, her pozisyonun yalnızca kendinden önceki pozisyonlara dikkat etmesine izin verir. Gelecekteki token'lar maskelenir.

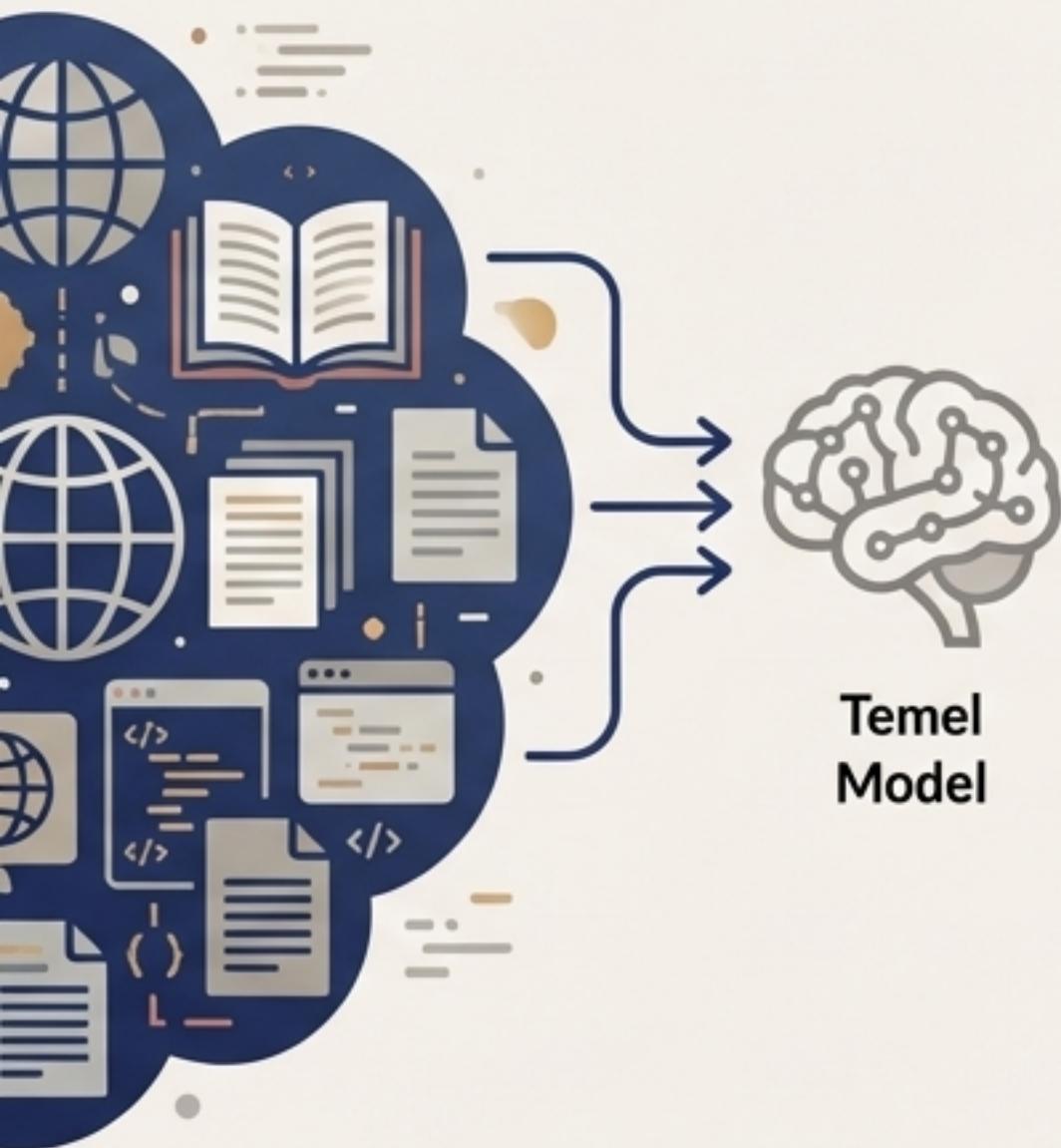
- **Neden?** Modelin eğitim sırasında cevabı 'görmesini' engellemek için. Amaç, bir sonraki kelimeyi tahmin etmeyi öğrenmektir, kopyalamayı değil. Bu, modelin tutarlı ve anlamlı metinler üretmesini sağlar.

Kullanım Alanları: Sohbet botları (ChatGPT), içerik üretimi, özetleme, kod yazma.



Bilgiyi Edinme ve Uzmanlaşma Süreci

1. Ön-Eğitim (Pre-training)

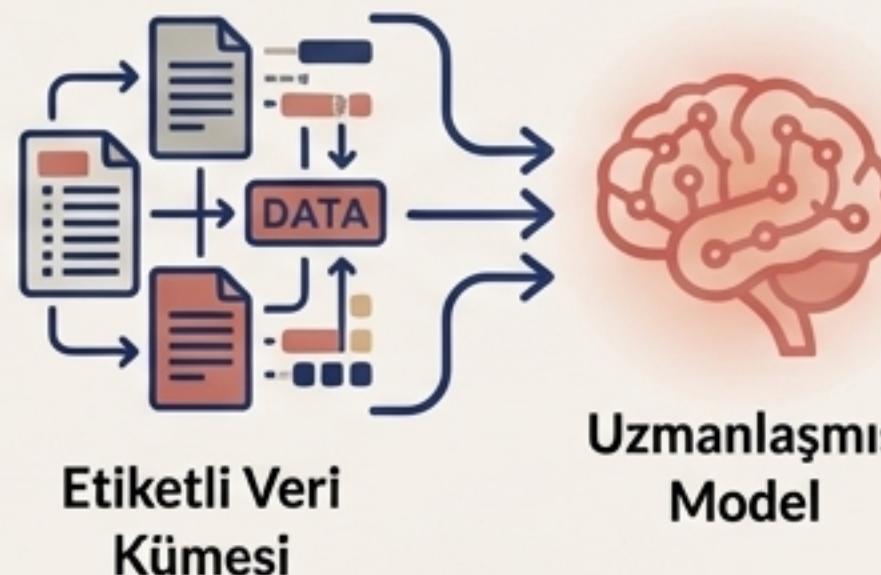


Ne? Model, internet ölçüğindeki devasa bir metin ve kod veriseti (trilyonlarca kelime) üzerinde eğitilir.

Amaç? Dilin genel kalıplarını, dilbilgisini, gerçek dünya bilgisini ve temel akıl yürütme becerilerini öğrenmek. Görev genellikle 'bir sonraki kelimeyi tahmin etmektir' (Decoder-Only için) veya 'maskelenmiş kelimeyi tahmin etmektir' (Encoder-Only için).

Sonuç: Geniş bir bilgiye sahip, ancak belirli bir görevde odaklanmamış bir temel model (foundation model).

2. İnce Ayar (Fine-tuning)



Ne? Ön-eğitimli temel model, daha küçük ve belirli bir görevde yönelik etiketli bir veri kümesi üzerinde tekrar eğitilir.

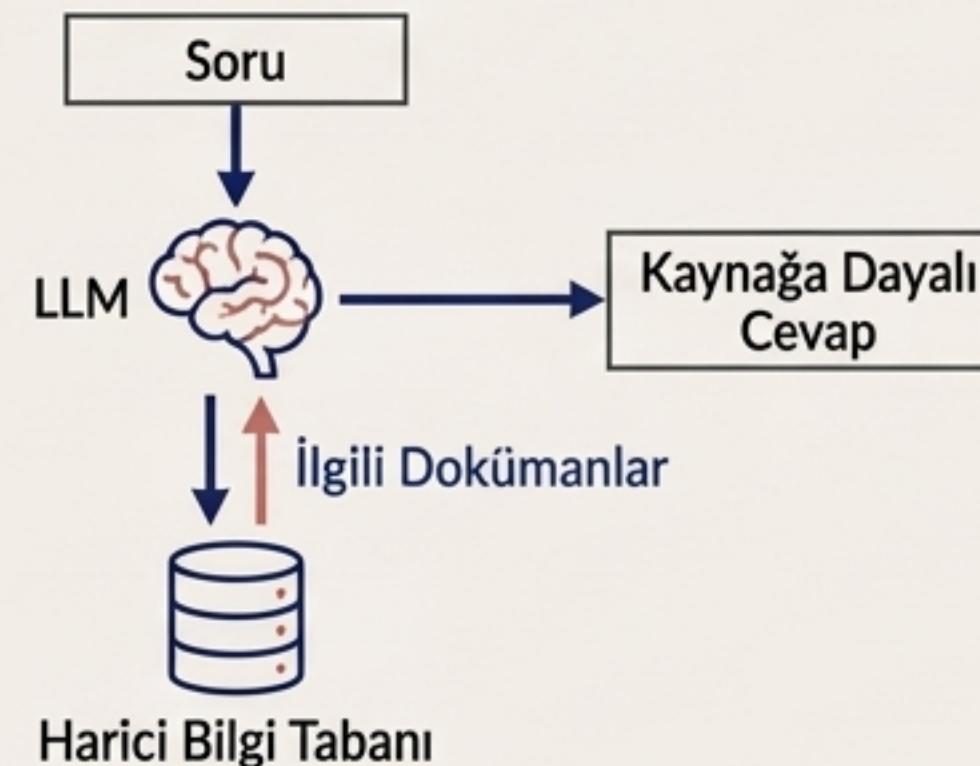
Amaç? Modelin genel yeteneklerini, belirli bir alanda (örneğin, müşteri hizmetleri diyalogları, hukuki belgeler, tıbbi raporlar) uzmanlaşması veya belirli bir davranış tarzını (örneğin, her zaman yardımsever bir asistan gibi cevap verme) benimsemesi için optimize etmek.

Sonuç: Belirli bir görev için yüksek performans gösteren **uzmanlaşmış bir model**.

Yetenekleri Geliştirmek: RAG ve Agent Kavramları

Temel modeller, harici bilgi ve araçlarla birleştirilerek yetenekleri katlanarak artırılabilir.

RAG (Retrieval-Augmented Generation / Geri-Getirme Destekli Üretim)

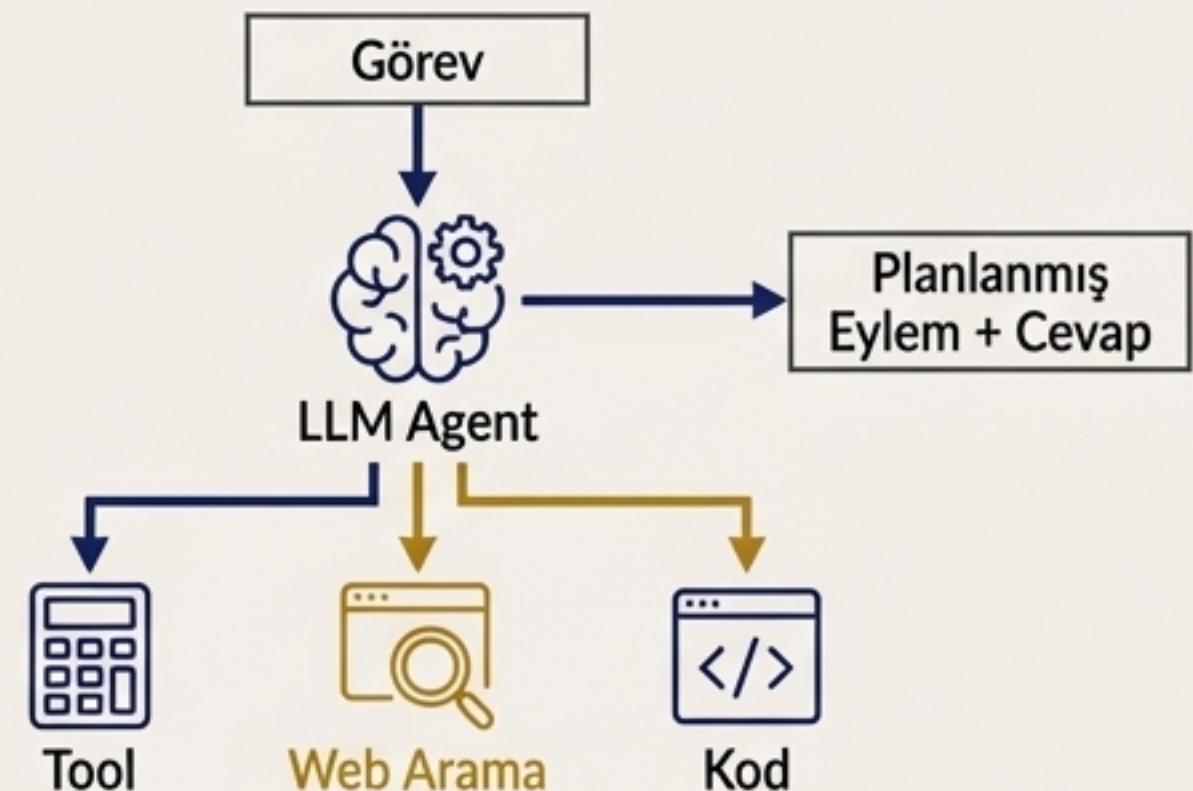


Model, bir soruya cevap vermeden önce, ilgili bilgiyi harici bir bilgi tabanından (örneğin, şirket içi dokümanlar, güncel haber veritabanı) 'geri getirir' (retrieve).

Cevabını bu güncel ve güvenilir bilgiye dayandırarak oluşturur.

Faydalari: Halüsinasyonları (uydurma bilgileri) azaltır, güncel verilerle çalışır ve kaynak göstermeye olanak tanır.

Agent'lar



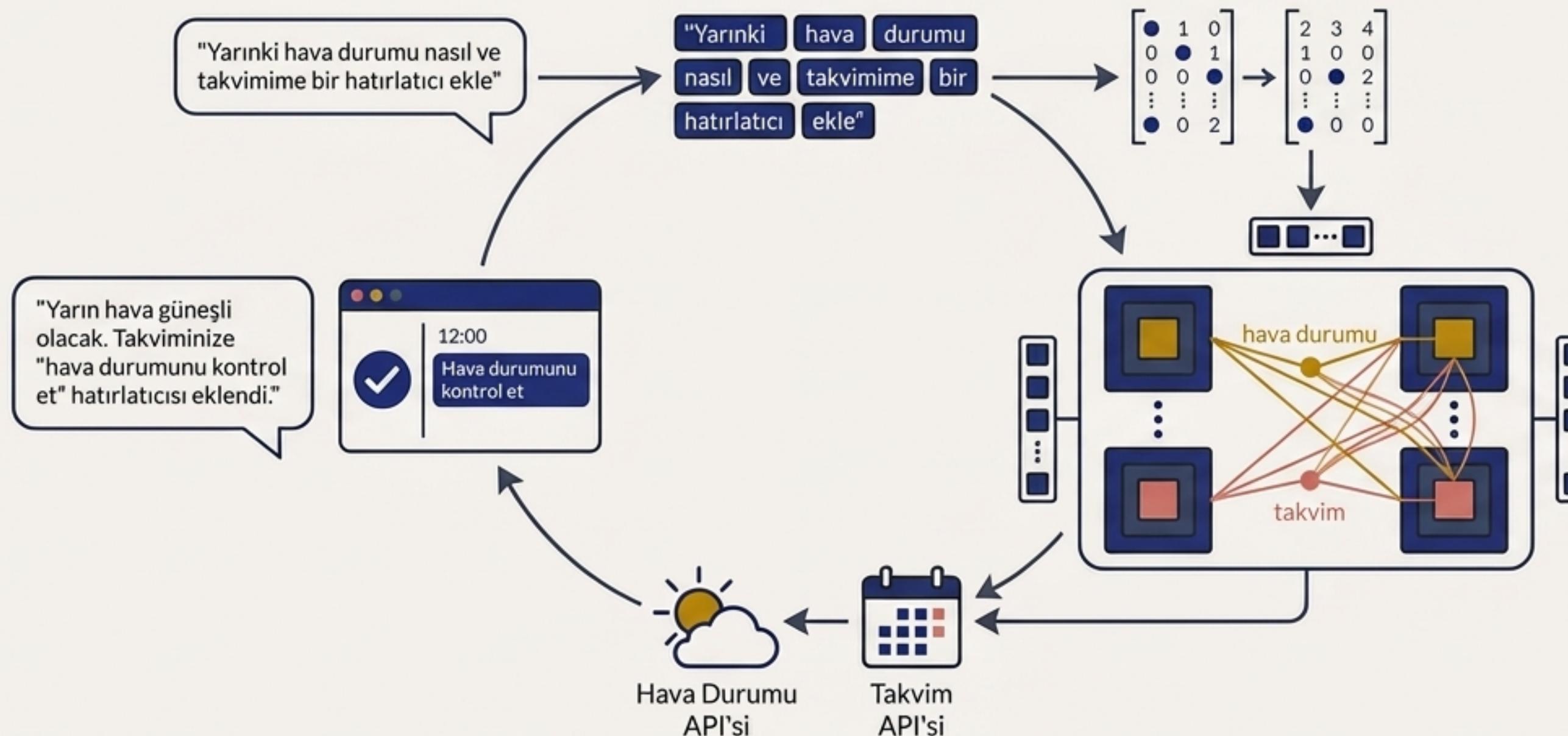
Model, sadece metin üremekle kalmaz, aynı zamanda 'araçlar' (tools) kullanabilir. Bu araçlar, bir hesap makinesi, bir arama motoru API'si veya bir kod yorumlayıcısı olabilir.

Model, bir görevi tanımlamak için hangi aracı ne zaman kullanacağına karar verir, aracın çıktısını yorumlar ve bir sonraki adımı planlar.

Sonuç: LLM, pasif bir metin üreticisinden, problem çözen ve eyleme geçen aktif bir 'agent'a dönüşür.

Yolculuğun Özeti: Bir Cümplenin Dönüşümü

Basit bir metinle başlayan yolculuk, dilin derinliklerinde anlam kazanan, farklı mimarilerde uzmanlaşan ve nihayetinde bilgiye erişip eyleme geçebilen sofistike bir yapıya dönüştü.



Bu dönüşüm, Transformer mimarisinin modern yapay zekayı nasıl şekillendirdiğinin en net göstergesidir.