

# DOĞAL DİL İŞLEME

## NLP (Nature Language Processing)

Mehmet Göktuğ Gökçe | 212923025

Rabia Durgut | 212923003

Aslı Şemşimoğlu | 212923001

Sinan Malak | 212923008

Bilgehan Bayrak | 212923009

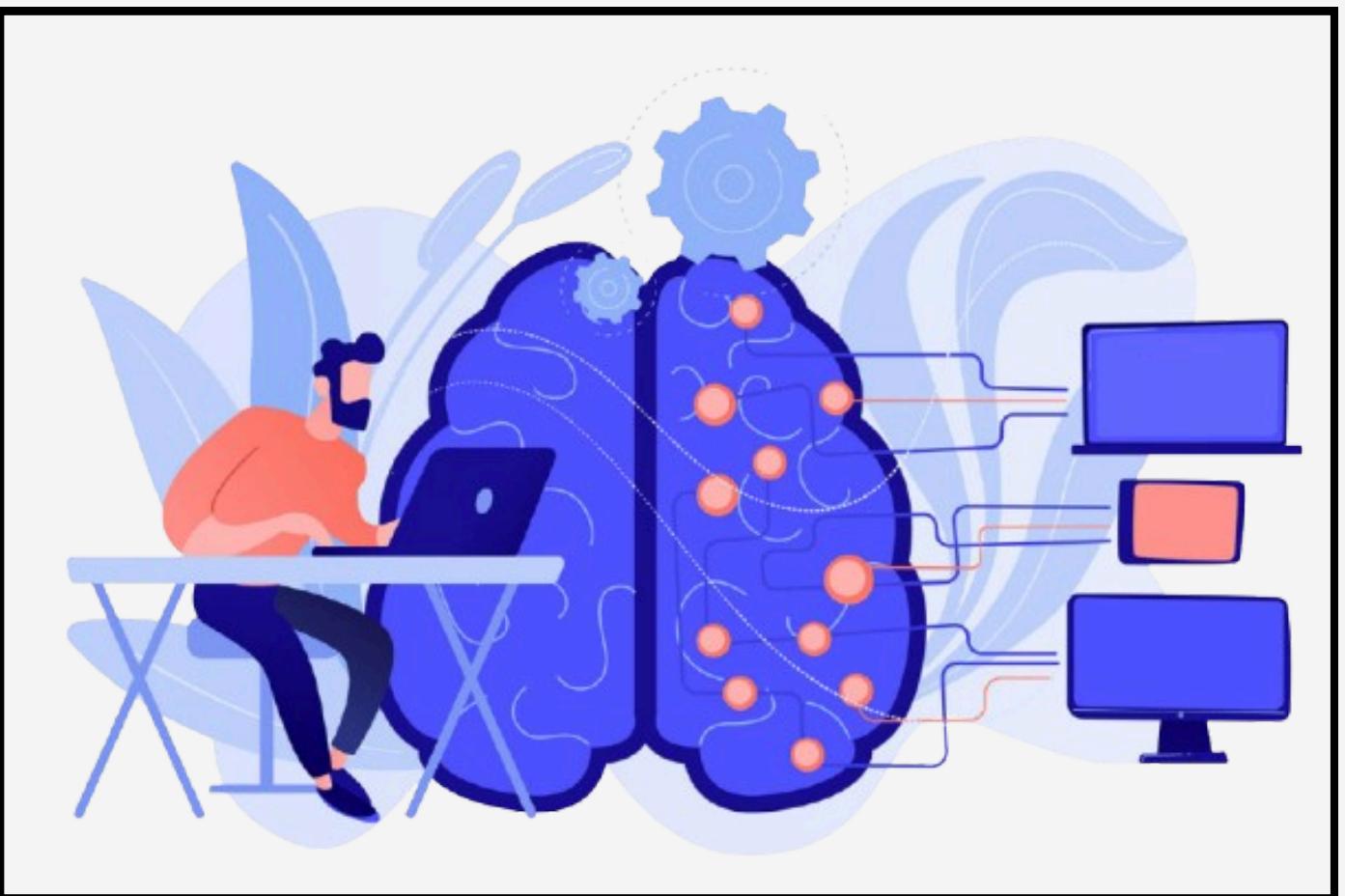
# SUNUM İÇERİĞİ

- 01 NLP'ye Genel Bakış**
- 02 Temel NLP Teknikleri**
- 03 Temel NLP Modelleri**
- 04 Attention Mekanizması**
- 05 Transformer Modelleri**
- 06 Transfer Öğrenme**
- 07 Büyük Dil Modelleri**
- 08 Fine-Tuning**

# NLP'YE GENEL BAKIŞ

Doğal Dil İşleme (NLP), bilgisayarların insan dilini anlama, yorumlama ve üretme yeteneğini geliştirmeye çalışan bir yapay zeka alanıdır.

*NLP, dilin yapısal ve anlamsal yönlerini analiz ederek, makinelerin yazılı veya sözlü dili anlamasını sağlar.*



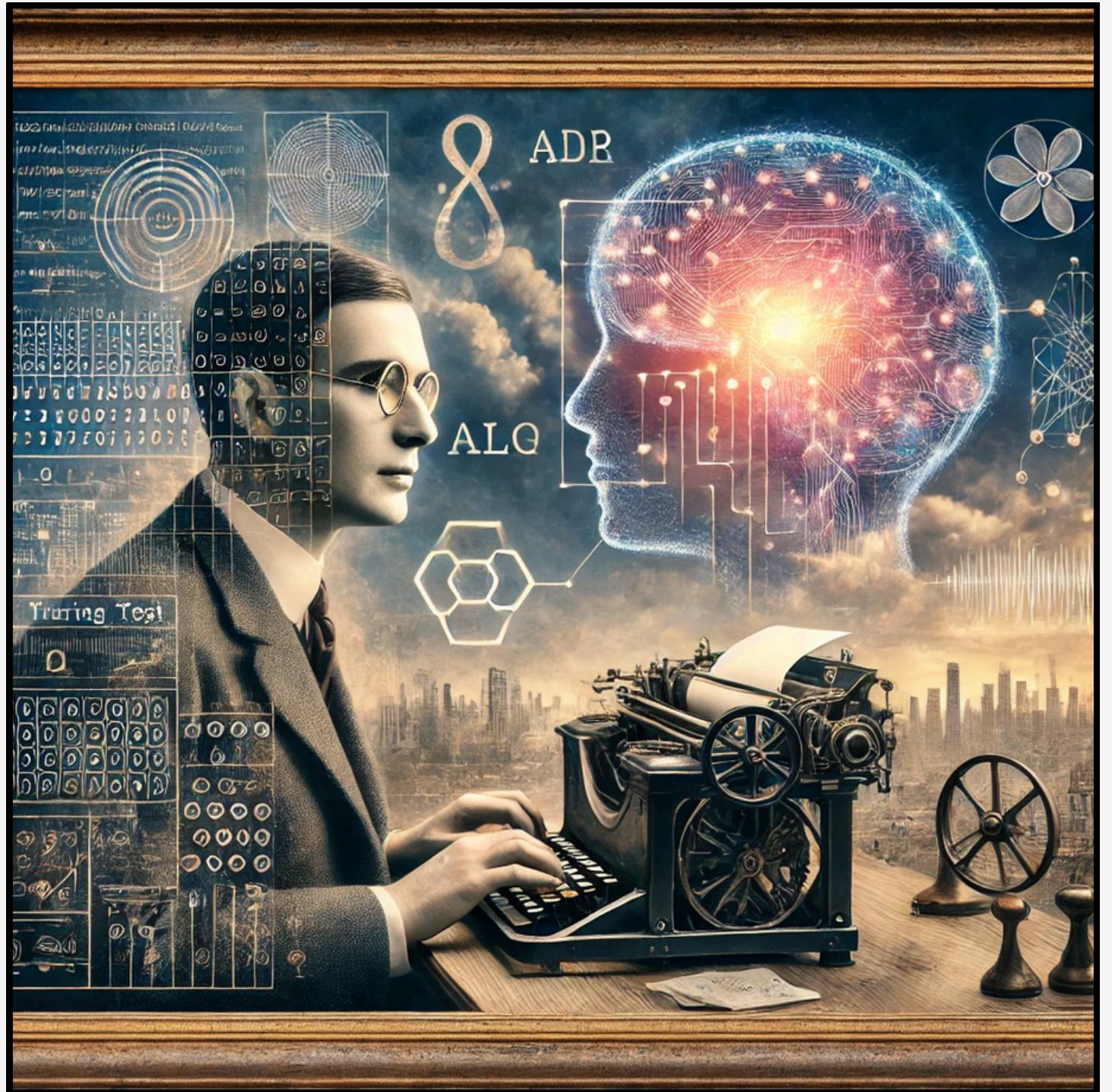
Görsel 1

# 1950'LERDE NLP

NLP'nin kökenleri, Alan Turing'in ünlü "**Turing Testi**" ile başlar.

Alan Turing, modern bilgisayar biliminin kurucusu olarak kabul edilir. 1950 yılında yazdığı "**Computing Machinery and Intelligence**" (**Bilgisayar Makineleri ve Zeka**) adlı makalesi, yapay zeka araştırmalarına yön veren önemli bir çalışmadır.

Turing bu makalede, makinelerin "düşünebilir" olup olamayacağını sorgulamış ve "**Turing Testi**"ni ortaya atmıştır.



# 1950'LERDE NLP

Turing Testi'nde, bir insan hakem, bir bilgisayar ve bir başka insanla iki ayrı ortamda yazılı iletişim kurar. Hakem, kimin insan, kimin makine olduğunu anlamaya çalışır. Eğer hakem, makinenin verdiği cevapları insanlardan ayırt edemezse, bu makinenin "**insan gibi düşünebildiği**" anlamına gelir.



# 1960'LARDA NLP

Bu dönemde, 1966 yılında Joseph Weizenbaum tarafından insan dilini işleyebilen ilk bilgisayar sistemi **ELIZA**, gelişmelerin en bilinen örneklerinden biridir.

Kullanıcı, metin olarak kendi düşüncelerini ve duygularını yazdığında, **ELIZA** bu girdilere önceden belirlenmiş kurallara göre yanıt veriyordu. Örneğin, eğer kullanıcı "**Kendimi kötü hissediyorum**" gibi bir cümle yazarsa, **ELIZA** bunun içindeki "**kötü**" kelimesini tanır ve "**Neden kendinizi kötü hissediyorsunuz?**" gibi bir yanıt verebilirdi.

# 1960'LARDA NLP

```
Welcome to
      EEEEEEE  LL    IIII    ZZZZZZ    AAAAAA
      EE       LL    II     ZZ     AA     AA
      EEEEEEE  LL    II     ZZZ    AAAAAAAA
      EE       LL    II     ZZ     AA     AA
      EEEEEEE  LLLLLL  IIII  ZZZZZZ    AA     AA

Eliza is a mock Rogerian psychotherapist .
The original program was described by Joseph Weizenbaum in 1966 .
This implementation by Norbert Landsteiner 2005 .

ELIZA: Is something troubling you ?
YOU: Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU: They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU: Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU: He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU: It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

## 1970-1980'LERDE NLP

1970'ler ve 1980'ler, NLP'de **kural tabanlı sistemlerin doruk noktasıydı.** Ancak, bu sistemlerin sınırlamaları, araştırmacıları daha esnek ve genellenebilir olan istatistiksel yaklaşımalar geliştirmeye yöneltti.

Bu sistemler, fiil çekimleri, isim tamlamaları veya cümle yapısı gibi dil bilgisel kuralların önceden tanımlanarak bilgisayarlarla öğretilmesi yoluyla çalışıyordu.

1980'lerin sonuna doğru **istatistiksel dil modelleri**, verinin gücünü kullanarak daha **büyük ve karmaşık** dil işleme sorunlarına çözüm sunmaya başladı.

# 1980-1990'LARDA NLP

## HMM (Hidden Markov Model)

Gizli Markov Modelleri, dizisel verilerle çalışan ve belirli bir gözlem dizisinden yola çıkarak gizli durumları modellemeyi amaçlayan istatistiksel bir yaklaşımdır.

HMM, bir durumlar zinciri oluşturur ve bu zincirdeki her adımda bir durumdan diğerine geçiş olasılıklarına göre bir geçiş yapılır. Her geçiş, bir gözlemle ilişkilendirilir. Sistem, gözlenen verilerden yola çıkarak en olası durumu tahmin eder.

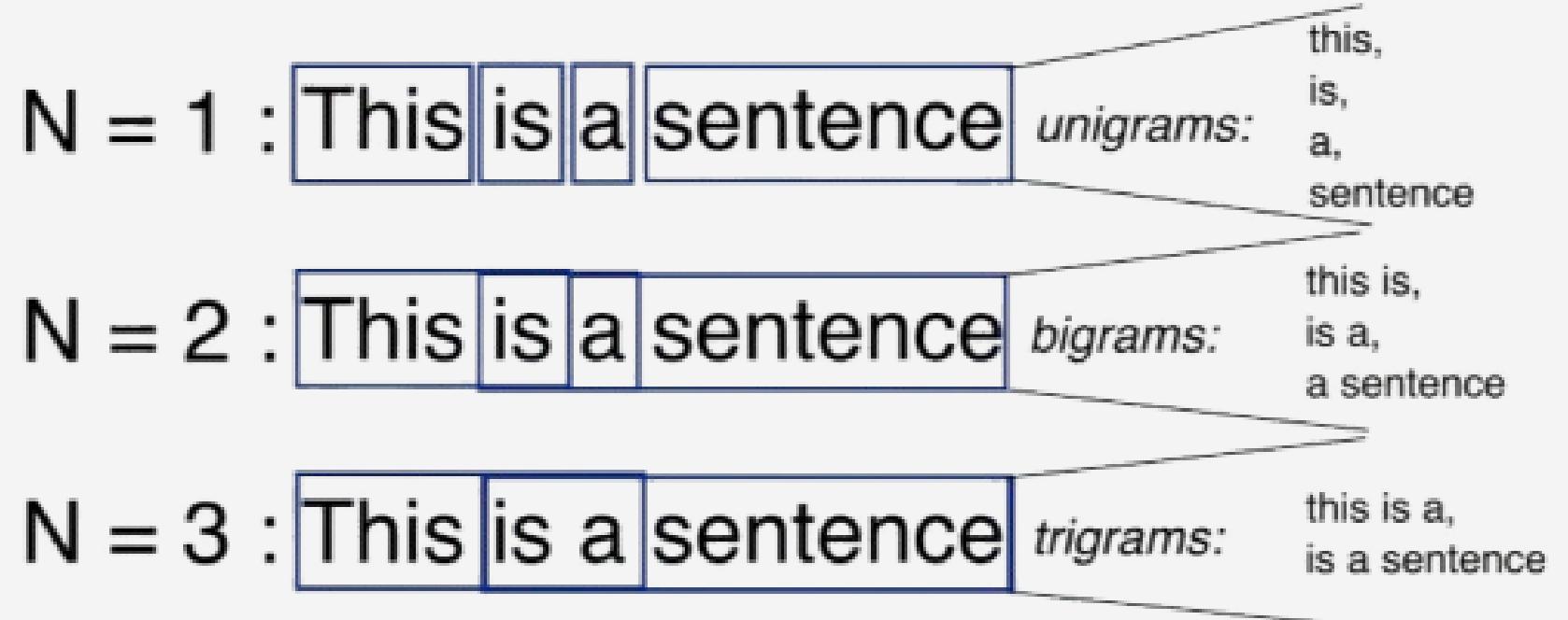
- Bugün yağmur yağıyorsa -> yarın yağmur yağma ihtimali = 0.4
- Bugün yağmur yağıyorsa -> yarın yağmur yağmama ihtimali = 0.6
- Bugün yağmur yağmıyorsa -> yarın yağmur yağma ihtimali = 0.2
- Bugün yağmur yağmıyorsa -> yarın yağmur yağmama ihtimali = 0.8

# 1990'LARDA NLP

Geliştirilen dil modelleri, büyük veri kümeleri kullanılarak eğitildi. Bu veri kümeleri, genellikle gerçek dünyadaki dil kullanımını içeren metinlerden oluşuyordu

Örneğin, N-gram modelleri, dilin olasılık dağılımlarını inceleyerek dildeki kelime dizilimlerinin istatistiksel yapısını anlamaya başladı.

n-Gram modelleri, dildeki kelimeler arasında ilişki kurarak gelecekteki kelimelerin olasılıklarını tahmin ediyordu. Bir trigram modeli, bir cümledeki üçüncü kelimenin, önceki iki kelimeye dayanarak tahmin edilmesini sağlar.



Görsel 3

# 1990'LARDA NLP

N-gram Modelleri: İstatistiksel dil modelleri genellikle N-gram modeline dayanır. Bu modellerde, bir kelimenin yalnızca önceki "n" adet kelimeye dayalı olarak belirli bir olasılıkla seçildiği varsayılar. En yaygın kullanılan N-gram modelleri:

- Unigram: Her kelimenin olasılığı yalnızca o kelimenin sıklığına bağlıdır.
- Bigram: Her kelimenin olasılığı yalnızca bir önceki kelimeye dayanır.
- Trigram: Her kelimenin olasılığı iki önceki kelimededen etkilenir.

N değeri arttıkça modelin bağlamsal anlamı yakalama yeteneği de artar, ancak bu daha büyük bir veri gereksinimi ve daha yüksek hesaplama maliyeti getirir.

# 1990'LARDA NLP

**Unigram:** Her kelimenin olasılığı bağımsız olarak hesaplanır. Kelimelerin cümledeki sırası veya öncesindeki kelimeler dikkate alınmaz.

Örneğin, elimizde şu cümlenin olduğunu düşünelim:

- Okuldan sonra sinemaya gitti. Eve gelmedi.

Bu durumda unigram model olasılık hesaplamaları için: “okuldan”, “sonra”, “sinemaya”, “gitti”, “eve”, “gelmedi” kelimelerini ele alır. Kendinden önceki kelimeye bağımsız olduğu için anlamsal bütünlüğü tam anlamıyla yakalayamayız.

# 1990'LARDA NLP

**Bigram:** Bigram modelinde, her kelimenin olasılığı yalnızca bir önceki kelimeye bağlıdır.

- Okuldan sonra sinemaya gitti. Eve gelmedi.

Bu durumda bigram model olasılık hesaplamaları için: “okuldan sonra”, “sonra sinemaya”, “sinemaya gitti”, “gitti eve”, “eve gelmedi” olarak ele alınır. Kendinden önceki bir kelimeye bağımlıdır.

# 1990'LARDA NLP

**Trigram:** Trigram modelinde, bir kelimenin olasılığı önceki iki kelimeye bağlıdır.

- Okuldan sonra sinemaya gitti. Eve gelmedi.

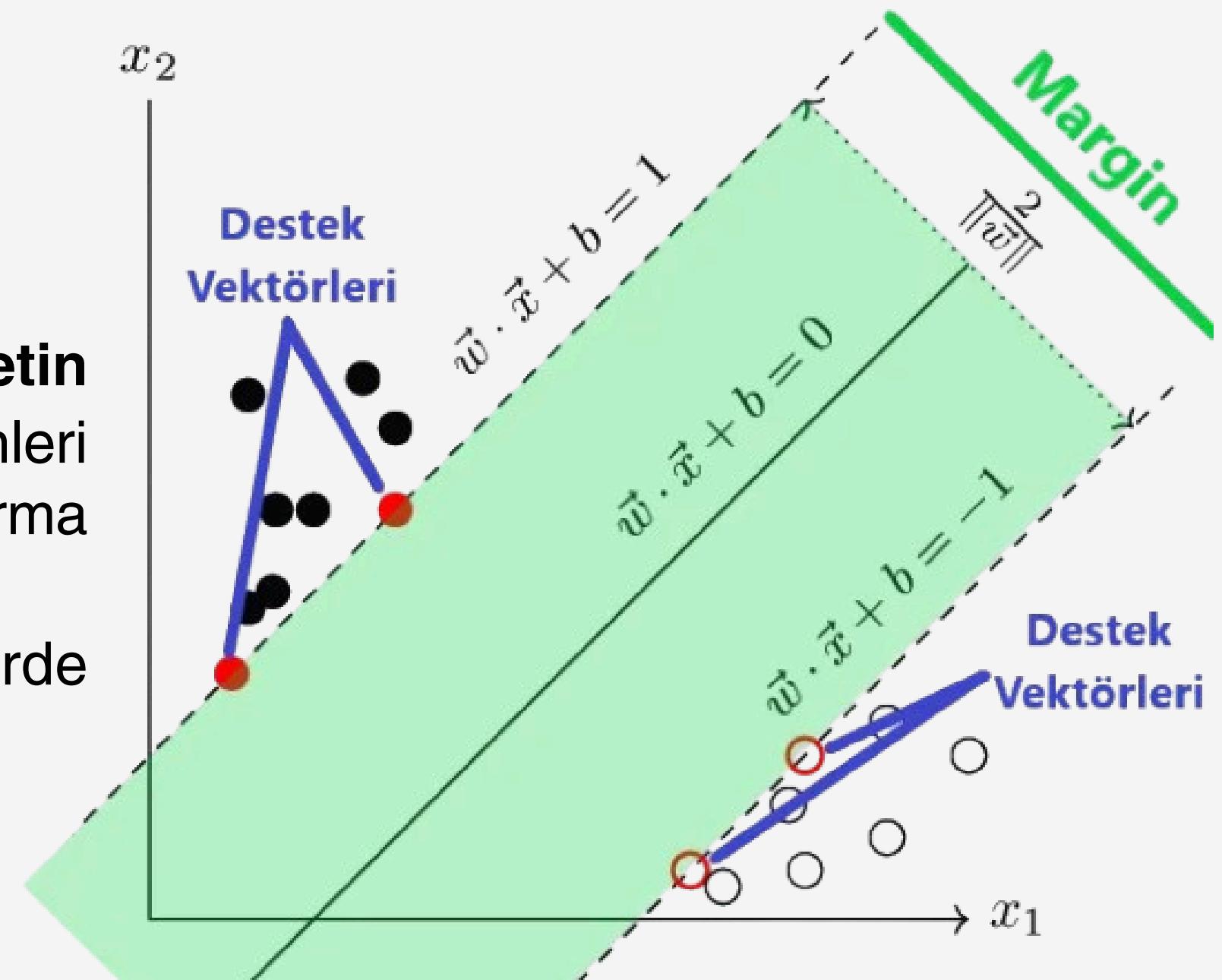
Bu durumda trigram model olasılık hesaplamaları için: “okuldan sonra sinemaya”, “sonra sinemaya gitti”, “sinemaya gitti eve”, “gitti eve gelmedi” olarak ele alır. Kendinden önceki iki kelimeye bağımlıdır.

# 1990'LARDA NLP

Metin	<b>“Okuldan sonra sinemaya gitti. Eve gelmedi.”</b>
Unigramlar	<b>“okuldan”, “sonra”, “sinemaya”, “gitti”, “eve”, “gelmedi”</b>
Bigram	<b>“okuldan sonra”, “sonra sinemaya”, “sinemaya gitti”, “gitti eve”, “eve gelmedi”</b>
Trigramlar	<b>“okuldan sonra sinemaya”, “sonra sinemaya gitti”, “sinemaya gitti eve”, “gitti eve gelmedi”</b>
N-gramlar (n=4)	<b>“okuldan sonra sinemaya gitti”, “sonra sinemaya gitti eve”, “sinemaya gitti eve gelmedi”</b>

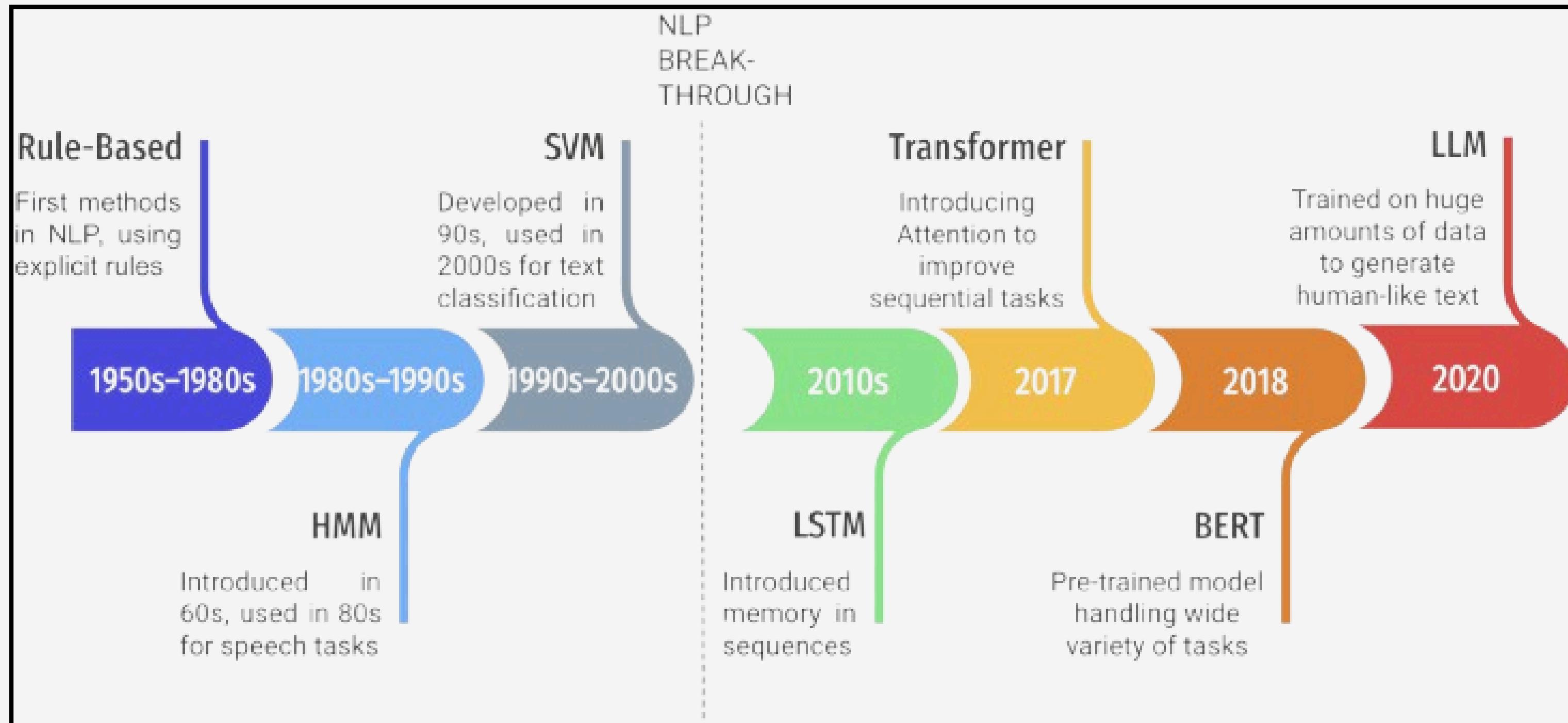
# 2000'LERDE NLP

Destek Vektör Makineleri, 2000'lerde **metin sınıflandırma** için kullanılmaya başlandı. **SVM**, metinleri çeşitli sınıflara ayırmak için güçlü bir sınıflandırma algoritmasıdır ve özellik vektörleri ile çalışır. Özellikle **spam e-posta tespiti** gibi görevlerde başarılı sonuçlar verdi.



Görsel 4

# NLP KIRILMA NOKTASI



Görsel 5

# 2010'LARDA NLP

Uzun Kısa Süreli Bellek (LSTM) hücreleri, RNN'lerin uzun süreli bağımlılıkları öğrenmedeki eksikliklerini gidermek için geliştirildi. LSTM'ler, zaman içinde bilgiyi tutma ve unutma kapasitelerine sahipti, bu da uzun dizilerde daha etkili öğrenme sağladı.

LSTM hücreleri, her bir adımda bilgiyi unutma, saklama ve çıkartma gibi işlemler yaparak, uzun vadeli bilgilerin tutulmasını sağlar. Bu yapılar sayesinde, geçmiş adımlardaki bilginin kaybolması engellenir.

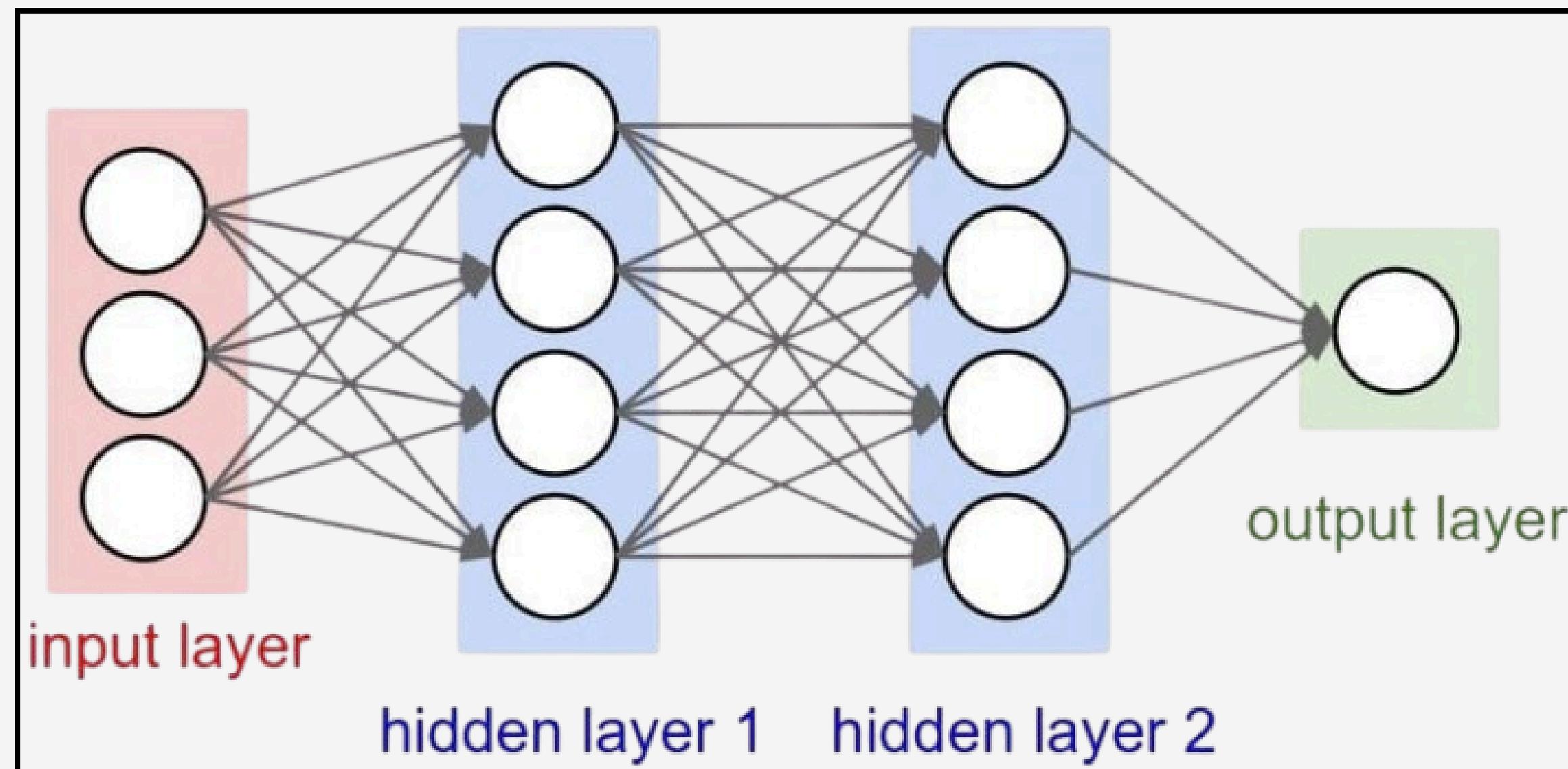
# 2010'LARDA NLP

Doğal Dil İşleme (NLP) alanında devrim niteliğinde bir değişim başladı. Bu değişim, büyük ölçüde derin öğrenme yöntemlerinin ve sinirsel ağların gelişimiyle mümkün oldu.

Derin öğrenme, makine öğrenmesinin bir alt dalıdır ve çok katmanlı yapay sinir ağları kullanılarak verilerin işlenmesini sağlar. NLP'de, bu derin sinir ağları sayesinde dilin daha karmaşık yapısı işlenebilir hale geldi.

# 2010'LARDA NLP

Yapay sinir ağları, bir dizi katmandan(layer) oluşur ve her katman verileri işleminden geçirerek daha derin bir anlam çıkarır. Bu sayede, dilin anlamı, yapısı ve bağlamı gibi özellikler daha iyi anlaşılabilir. Sinirsel ağlar, özellikle dilin ardışık yapısına (kelimelerin sıralanışı) ve bağlamına dayalı kararlar verir.



Görsel 6

# 2010'LARDA NLP

2017'de **Transformer** mimarisi tanıtıldı ve NLP'de büyük bir sıçrama yaşandı. Dikkat (attention) mekanizmasıyla çalışan bu modeller, bir cümledeki kelimelerin birbirleriyle olan ilişkilerini çok daha esnek ve etkili bir şekilde analiz edebiliyordu. Transformer modelleri, özellikle uzun metinlerdeki anlam ilişkilerini daha iyi anladığı için dil işleme performansında önemli bir iyileşme sağladı.

Derin öğrenmenin sunduğu bu yeni araçlarla birlikte büyük dil modelleri geliştirildi. Bu modeller, devasa veri kümeleri üzerinde eğitildi ve insanların dil kullanımını daha iyi anlamaya başladı. Özellikle GPT ve BERT gibi modeller, NLP'nin günümüzde olduğu noktayı simgeler.

# 2010'LARDA NLP

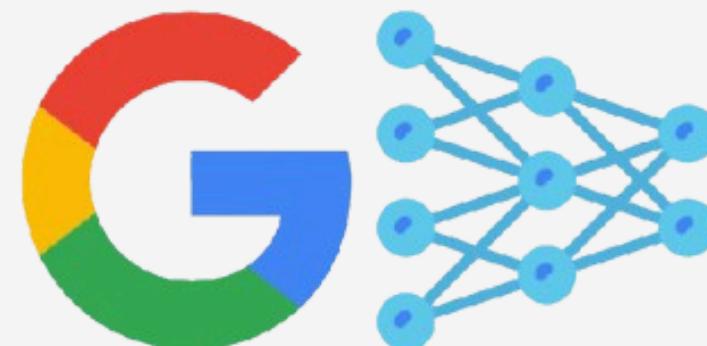
GPT (Generative Pre-trained Transformer) modeli, OpenAI tarafından geliştirilen bir dil modelidir. GPT'nin en önemli özelliği, önceden devasa metin veri kümeleri üzerinde eğitilmesi ve bu sayede dilin kurallarını, bağlamlarını ve ilişkilerini öğrenmesidir.

## OpenAI Modellerinin Gelişim Süreci

MODEL	PARAMETRE SAYISI	ÇIKIŞ TARİHİ
GPT-2	1.5 Milyar	2019
GPT-3	175 Milyar	2020
GPT-3.5	175 Milyar (Optimize)	2023 Başları
GPT-4	Tahmini 1 trilyon	2024
GPT-4 Turbo	Belirtilmedi	2023 Sonu
GPT o1-Preview	Belirtilmedi	2024 Ortaları

# 2010'LARDA NLP

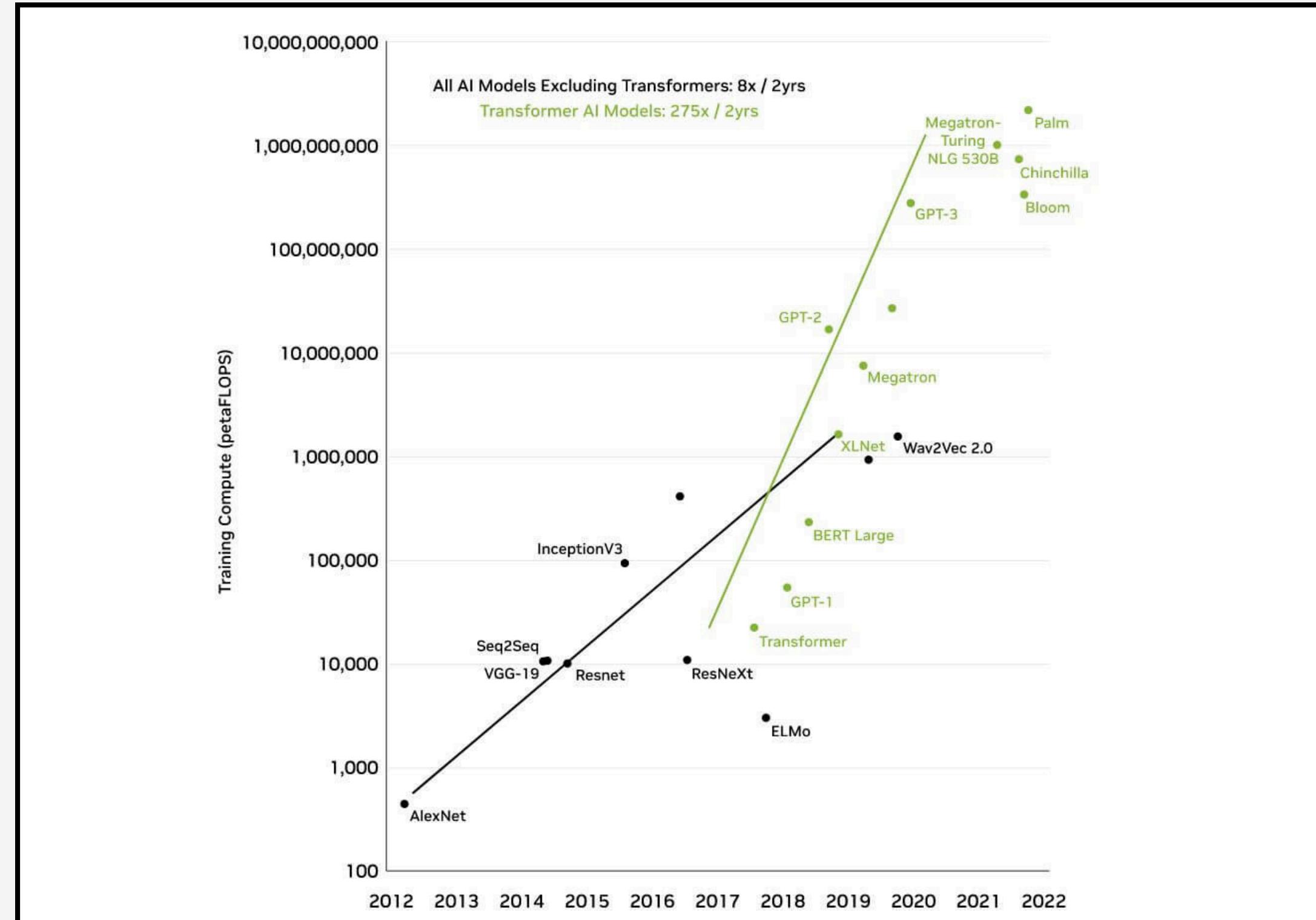
**BERT**, 2018 yılında **Google** tarafından geliştirilen bir başka devrim niteliğindeki dil modelidir. BERT'in en büyük farkı, dilin her iki yönünü (ileri ve geri) birden öğrenebilmesidir. Bu, cümledeki kelimelerin hem önceki hem de sonraki kelimelerle olan ilişkisini anlamayı mümkün kılar.



Google BERT Algorithm

Görsel 7

# 2010'LARDA NLP



Görsel 8

# Derin Öğrenme ve Büyük Dil Modellerinin Başarı Alanları

01

**Öğrenme Kapasitesi:** Derin öğrenme, dil modellerine **milyonlarca parametreyle karmaşık ilişkileri ve kalıpları öğrenme** kapasitesi kazandırır. Bu sayede, metin içinde daha anlamlı ve **bağlantılı tahminlerde** bulunabilirler.

02

**Bağlam Anlayışı:** Derin öğrenmenin katkısıyla LLM'ler, kelimeler arasındaki uzun mesafeli bağıntıları öğrenerek metnin daha geniş bir bağlamını anlamada ustalaşır. **Transformer** tabanlı modeller, bu sayede bağlamsal ilişkileri güçlü bir şekilde yakalayarak **tutarlı cevaplar** üretebilir.

# Derin Öğrenme ve Büyük Dil Modellerinin Başarı Alanları

03

**Metin Özeti:** Büyük dil modelleri, uzun metinleri özetleme konusunda da oldukça başarılı. Bu sayede haber metinlerinden, makalelerden veya belgelerden **otomatik olarak özet çıkarmak** mümkün hale geldi.

04

**Soru-Cevap Sistemleri:** BERT gibi modeller, arama motorları ve bilgi edinme sistemlerinde soruları daha iyi anlayarak doğru cevaplar sunmaya yardımcı olur. Örneğin, **Google arama motoru**, kullanıcıların sorularına daha alakalı cevaplar sunmak için BERT'i kullanır.

# NLP'İN TEMEL TEKNİKLERİ

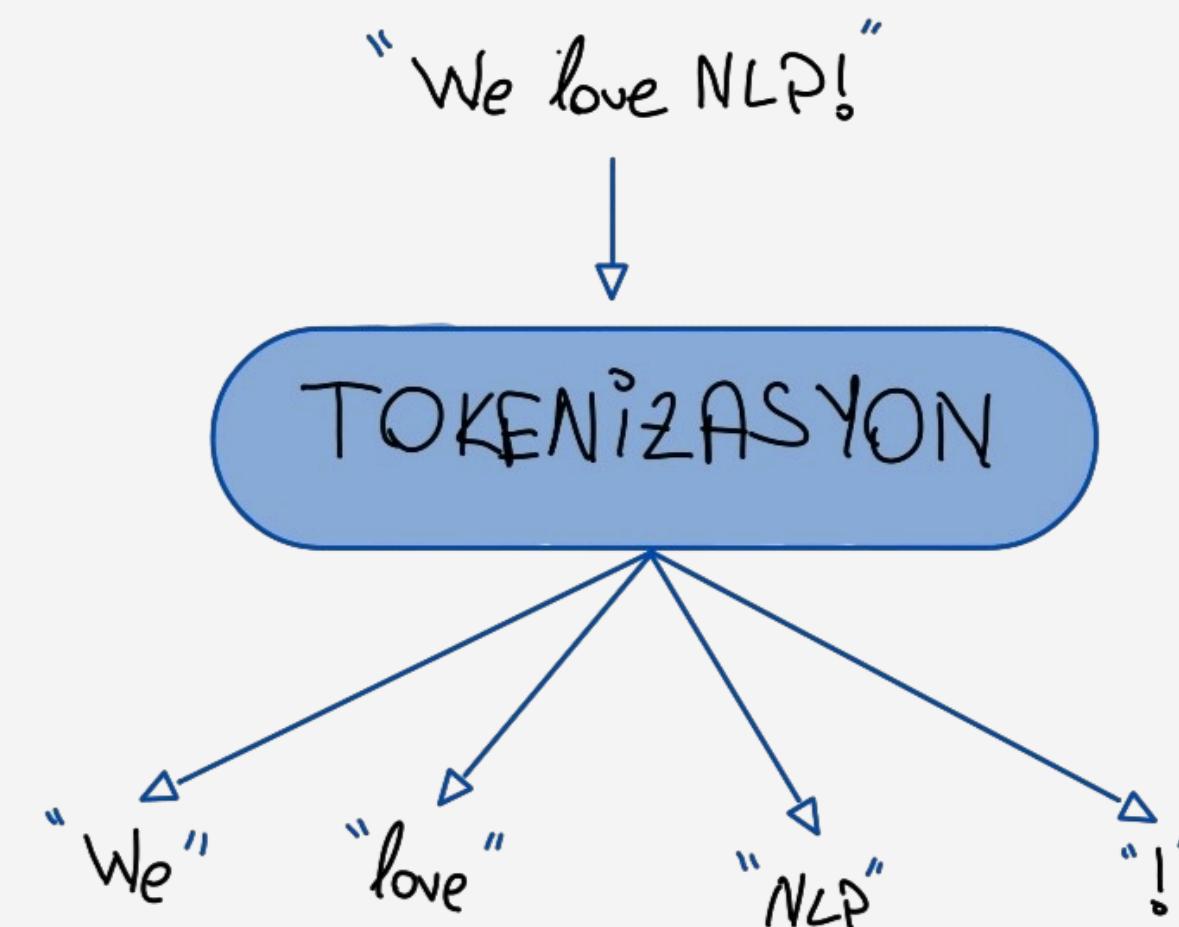
Doğal dil işleme (NLP) teknikleri ve modelleri, **insan dilini anlamak, işlemek ve üretmek** amacıyla kullanılır. Bu alandaki teknikler, hem temel işleme adımları hem de daha karmaşık modeller ve yaklaşımlar etrafında gelişmiştir. İşte NLP'de temel teknikler ve popüler modeller:

Text-To-Speech  
Tokenizasyon  
**Sentiment Analizi**  
Adlandırılmış Varlık Tanıma  
Speech-To-Text  
Stemming TF TF-IDF Puanı  
Lemmatization  
Stop-Word Filtreleme  
Part of Speech Etiketleme  
Bag Of Words IDF

# NLP'İN TEMEL TEKNİKLERİ

**Tokenizasyon:** Metnin **kelime** ya da **cümlelere bölünmesi** işlemidir. Bu, metni daha küçük parçalara ayırarak analiz etmeyi sağlar.

- **Kelime Tokenizasyonu:** Metni kelimelere böler.
- **Cümle Tokenizasyonu:** Metni cümlelere böler.



# NLP'İN TEMEL TEKNİKLERİ

**Stopword Filtreleme:** Anlam taşımayan ya da sık kullanılan kelimelerin (örneğin "ve", "ama") metinden çıkarılmasıdır. Bu adım, analiz performansını artırmak için gereklidir.

## Lemmatizasyon ve Gövdeleme:

- **Lemmatizasyon:** Kelimelerin sözlük biçimine indirgenmesidir. Örneğin, "koşuyor" kelimesi "koşmak" olarak lemmatize edilir.
- **Gövdeleme:** Kelimenin köküne indirgenmesidir. Örneğin, "koşuyor" kelimesi "koş" köküne indirgenebilir.

**LEMMATİZASYON:**   
Çiçeklik  
Çiçeklik -ler

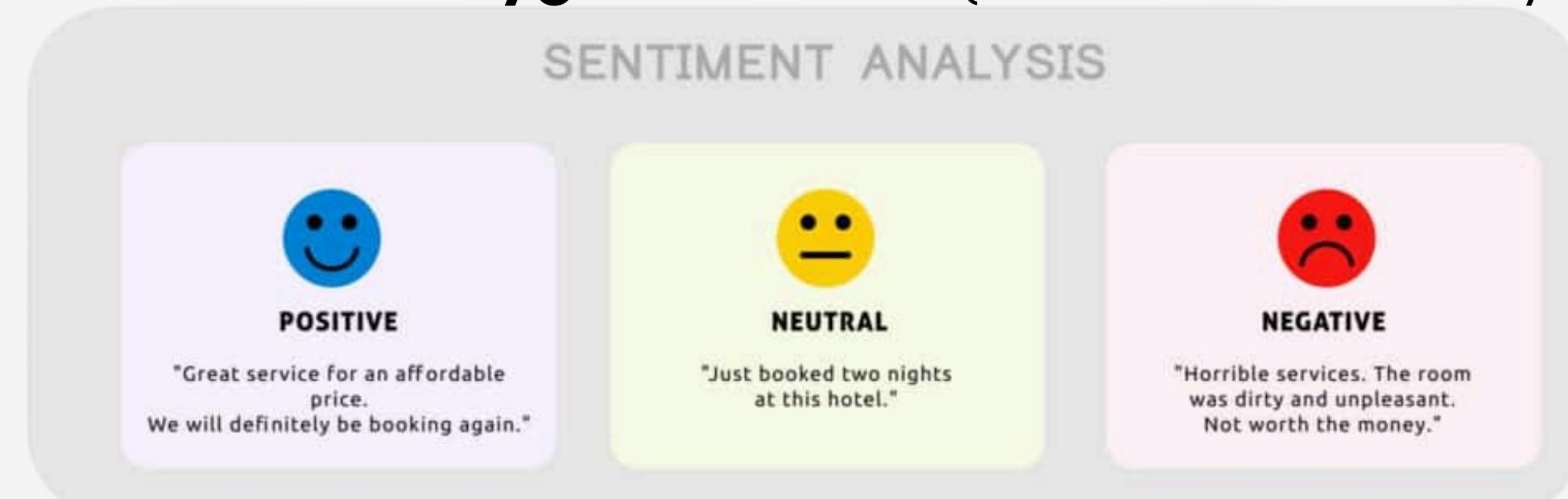
**GÖVDELEME:**   
Çiçeklik  
Çiçek -lik -ler

# NLP'İN TEMEL TEKNİKLERİ

**Adlandırılmış Varlık Tanıma (Named Entity Recognition, NER):** Metinlerde kişi adları, organizasyonlar, tarihler, yer isimleri gibi belirli varlıklarını tanıma işlemidir.

**Part of Speech (POS) Etiketleme:** Metindeki kelimelere dilbilgisel rollerini (isim, fiil, sıfat vb.) atama işlemidir.

**Sentiment Analizi:** Metnin **duygusal tonunu** (olumlu, olumsuz veya nötr) belirler.



Görsel 9

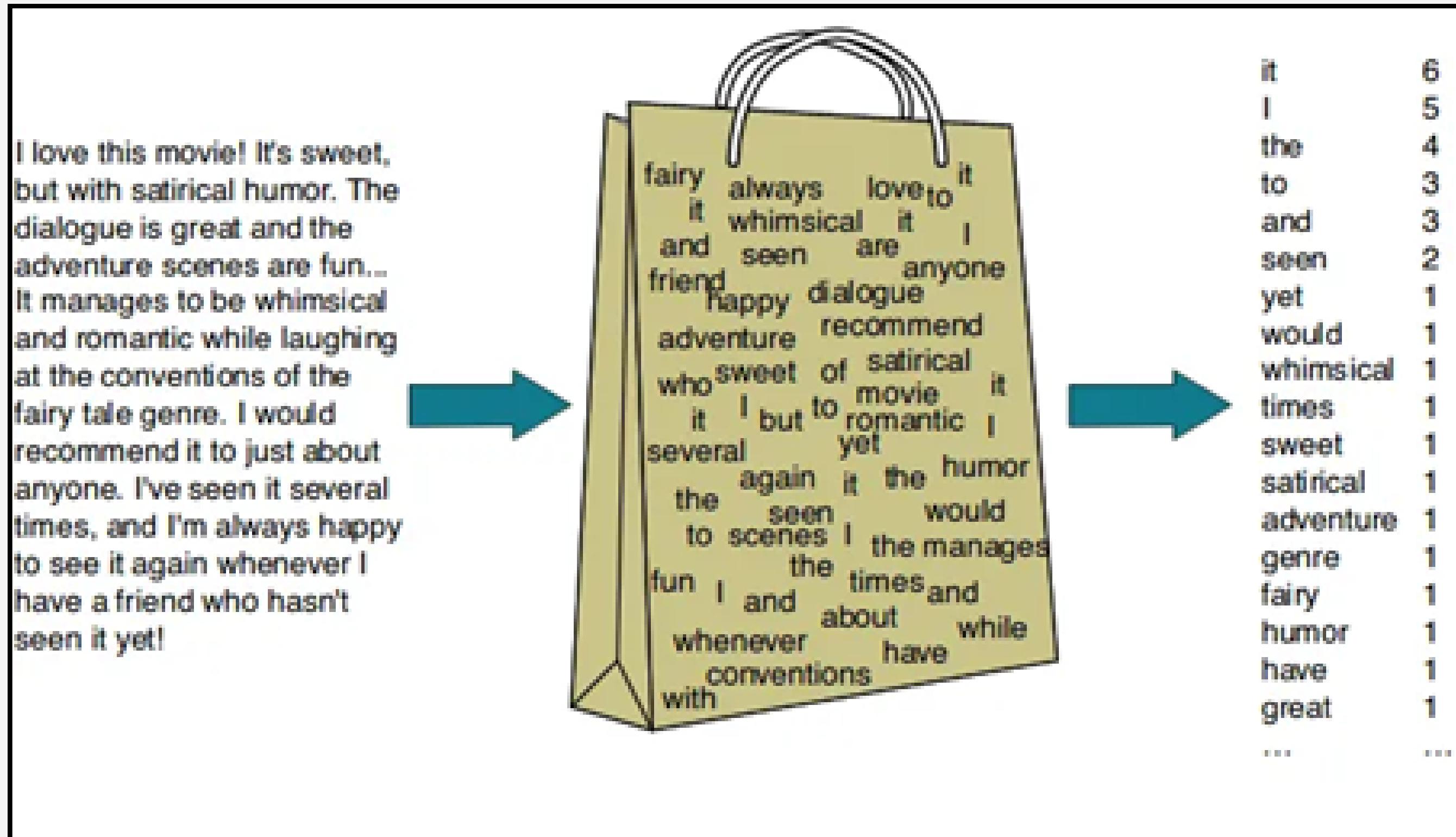
# TEMEL NLP MODELLERİ

**1. Bag Of Words:** Bag of Words, **kelimelerin sıklığına dayalı** olarak metni temsil eden basit bir yöntemdir. Her bir belge, tüm kelimelerin bir listesinden oluşan bir çantaya (bag) dönüştürülür.

! BoW modelinde, **kelimelerin sırası veya grameri dikkate alınmaz**, sadece kelimenin belgede bulunup bulunmadığı veya kaç kez geçtiği önemlidir.

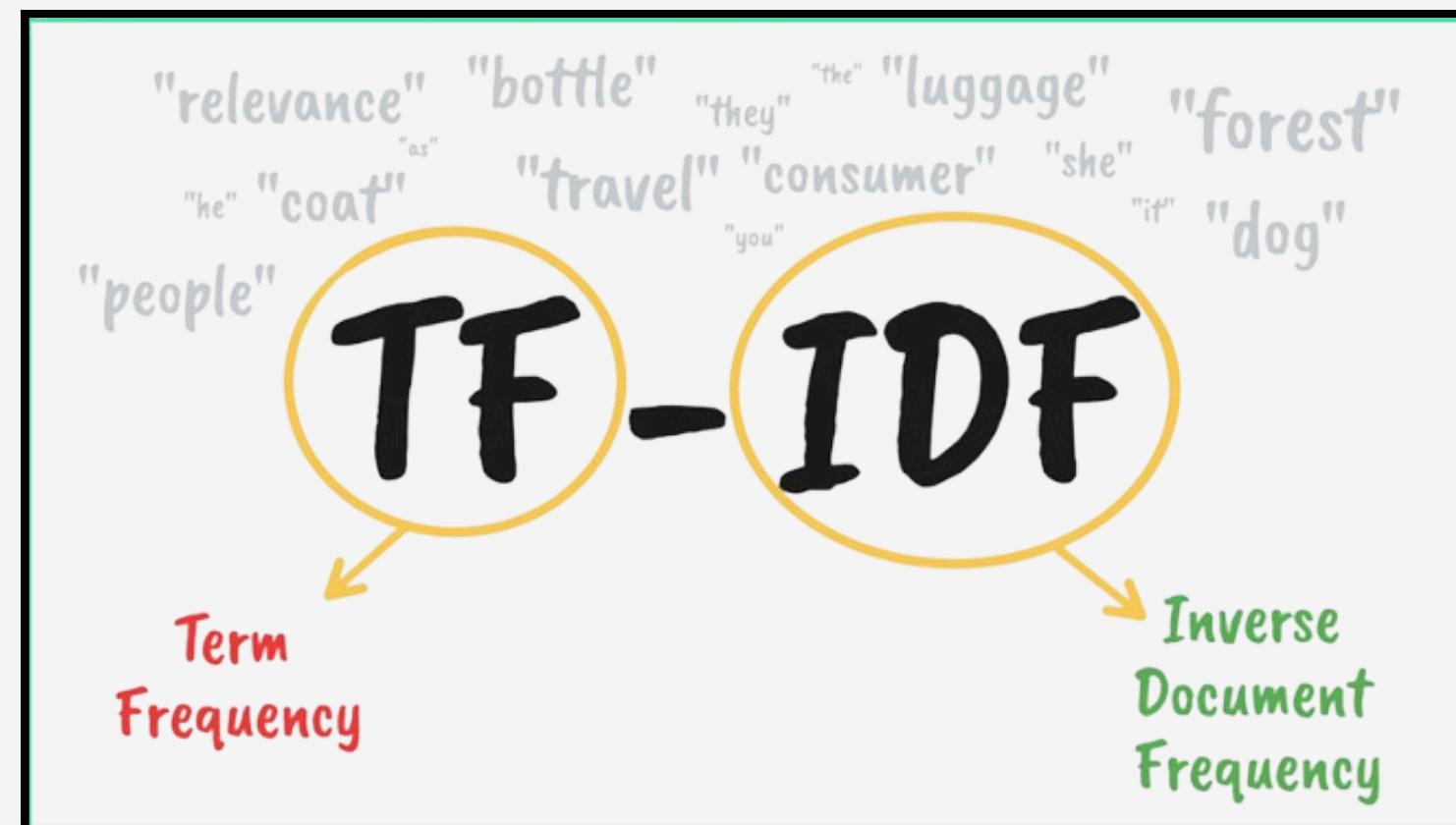
Bu yöntem, **metin sınıflandırma ve duygusal analizi** gibi NLP görevlerinde sıkça kullanılır. Ancak, metindeki bağlamı göz ardı eder ve bu, anlamı tam olarak yakalayamamasına neden olabilir. Yani bag of words modeli karmaşık görevler için çok iyi çalışmaz ancak basit sınıflandırma modelleri için kullanım kolaylığı ve basitliği nedeniyle sıklıkla tercih edilir.

# TEMEL NLP MODELLERİ



# TEMEL NLP MODELLERİ

**2. TF-IDF:** Bir kelimenin bir belge için ne kadar önemli olduğunu yansıtan **sayısal istatistik**tir. Doğal Dil İşleme (NLP) bağlamında, **TF-IDF** genellikle **metin madenciliği, bilgi alma ve metin kategorizasyonu** için kullanılır. Bir belgedeki önemli sözcükleri daha büyük bir külliyata göre tanımlamaya yardımcı olarak, belirli bir bağlamdaki terimlerin benzersizliğini ve önemini yakalamanıza olanak tanır.



Görsel 11

# TEMEL NLP MODELLERİ

**TF (Term frequency):** Bir kelimenin bir belgede **ne kadar sıkılıkta geçtiğini** ölçer. Kelime ne kadar **sık olursa, TF o kadar yüksek** olur.

$$TF(t,d) = t \text{ kelimesinin } d \text{ belgesinde görünme sayısı} / d \text{ belgesindeki toplam terim sayısı}$$

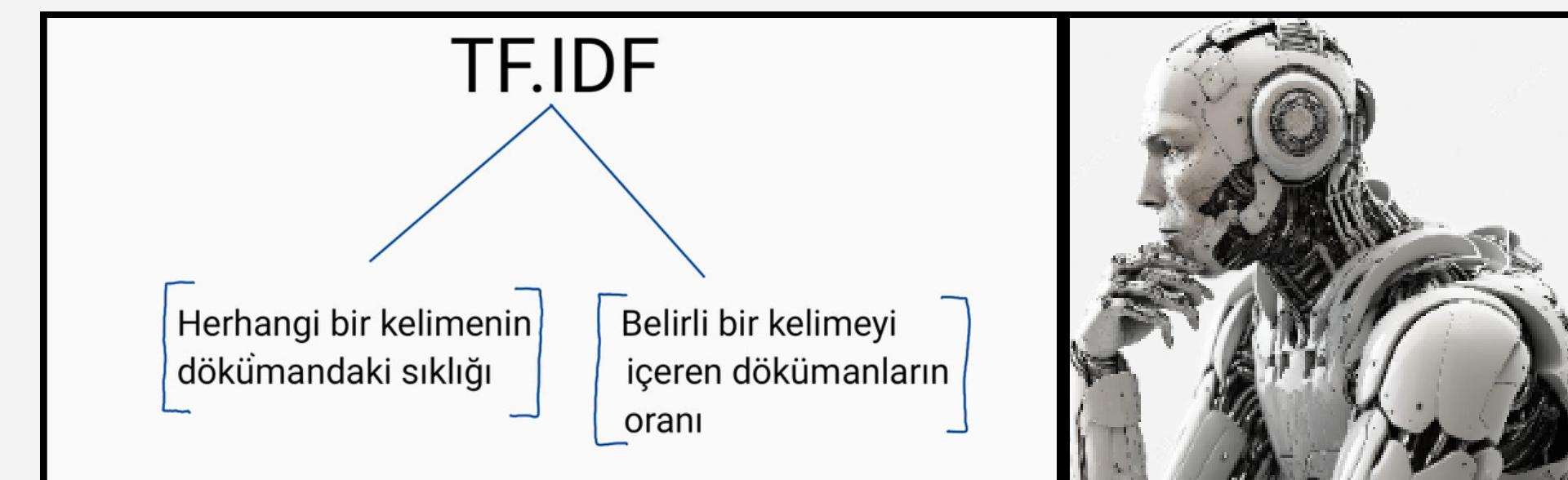
**IDF (Inverse Document Frequency):** Kelimenin, veri kümesindeki belgelerde **ne kadar nadir olduğunu** ölçer. Kelime ne kadar **nadir olursa IDF o kadar yüksek** olur.

$$IDF(t,D) = \log(D \text{ derlemesindeki toplam belge sayısı} / t \text{ terimini içeren belge sayısı})$$

**TF-IDF Puanı:** Bu, **TF** ve **IDF**'nin çarpımıdır ve size belirli bir belgedeki her terim(kelime) için bir **ağırlık** verir.

**TF-IDF** puanı ne kadar yüksekse, terim o belge için o kadar önemlidir.

$$TF-IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

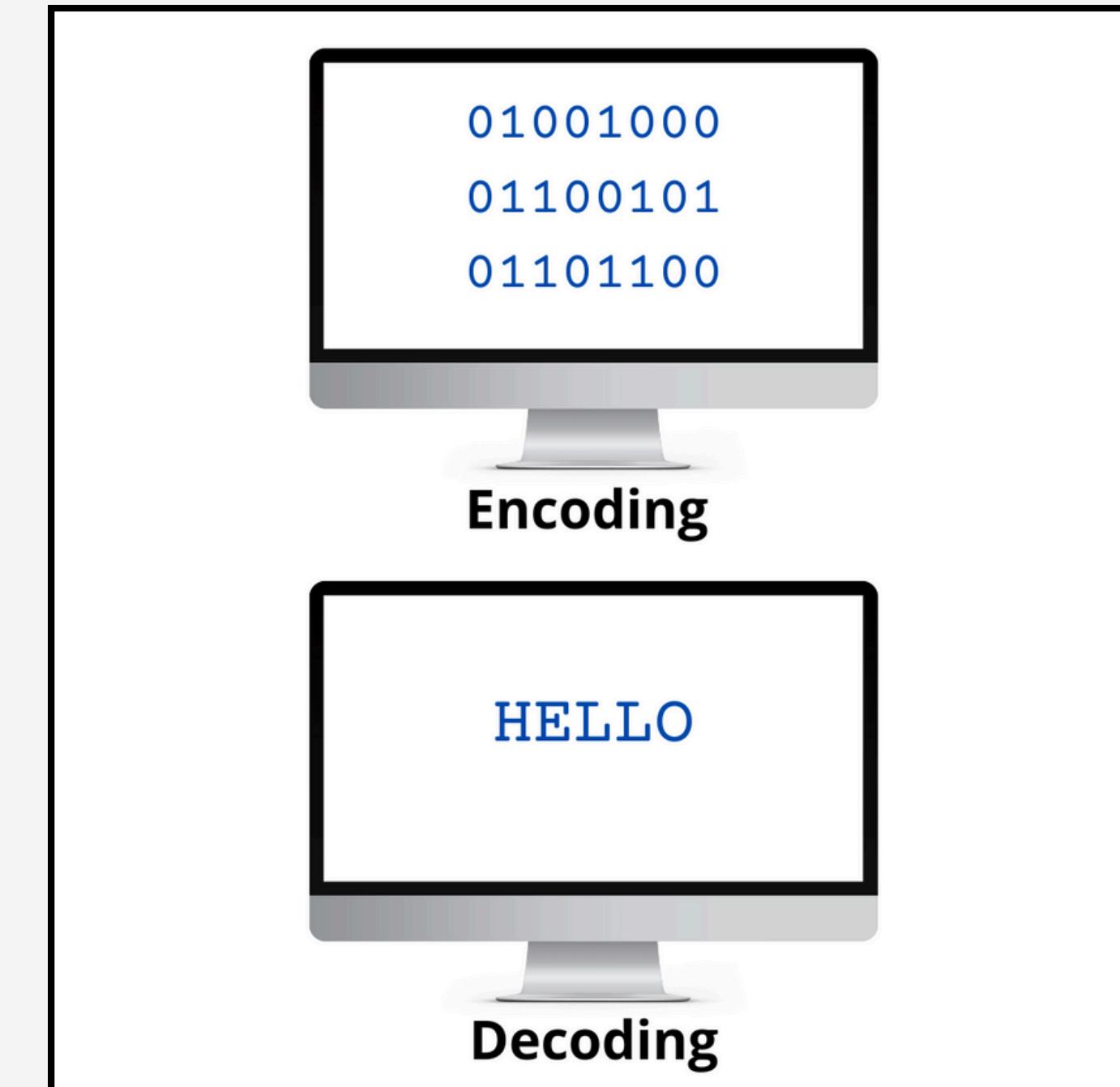


Görsel 12

# TEMEL NLP MODELLERİ

**ENCODİNG:** Encoding(Metinleri kodlamak), tüm doğal dil uygulamalarındaki en önemli adımlardan biridir. Encoding yöntemleri, kelimelerin, cümlelerin ya da diğer dil birimlerinin sayısal temsillerini oluşturur ve bu sayede makine öğrenimi algoritmalarının üzerinde çalışabileceği biçimde sokar.

Sinir ağları karakterler, kelimeler ve cümleler üzerinde değil; **sayılar, vektörler ve tensörler üzerinde** çalıştığından, bu özellikle derin öğrenmede kritik öneme sahiptir.



Görsel 13

# TEMEL NLP MODELLERİ

**One-hot encoding:** Her kelimeyi, **kelimenin bulunduğu pozisyon için "1", diğer tüm pozisyonlar için "0" olan vektörlerle** temsil eder. Örneğin, "elma", "armut", "kiraz" kelimelerini içeren bir kelime kümesinde, "elma" şu şekilde temsil edilir: [1, 0, 0], "armut": [0, 1, 0] ve "kiraz": [0, 0, 1].

**Avantajları:** One Hot encoding **basit** ve **anlaşılır** bir yöntemdir

**Dezavantajları:** Bu yöntemde kelimeler arasındaki anlam ilişkileri veya benzerlikler temsil edilemez. Ayrıca, kelime sayısı büyüdükçe vektör boyutu artar ve bu durum bellekte büyük yer kaplamasına neden olur (sparse vektörler).

# TEMEL NLP MODELLERİ

**Örnek:**

Enflasyon, bir ekonomide mal ve hizmetlerin genel fiyat düzeyinde sürekli bir artış yaşanmasıdır. Bu durum, para biriminin satın alma gücünü azaltır ve tüketicilerin aynı miktarda mal veya hizmet almak için daha fazla para ödemeye yol açar. Enflasyonun yükselmesinin farklı nedenleri olabilir; maliyet enflasyonu (üretim maliyetlerinin artması), talep enflasyonu (talebin arzdan fazla olması) ve para arzındaki genişleme bunlar arasında sayılabilir. Enflasyonun makul düzeyde olması, ekonomik büyümeye teşvik sağlarken, yüksek enflasyon bireylerin birikimlerini eritebilir ve gelir dağılımında adaletsizlik yaratabilir. Bu nedenle, merkez bankaları ve hükümetler, enflasyonu kontrol altına almak amacıyla para politikaları (örneğin, faiz oranlarını değiştirmeye) ve mali politikalar uygular.

Kelime	One-Hot-Encoding
Enflasyon	1
Spor	0
Hizmet	1

# TEMEL NLP MODELLERİ

**Label Encoding:** Label encoding, doğal dil işleme (NLP) ve makine öğreniminde **kategorik verilerin sayısal değerlere** dönüştürülmesinde kullanılan bir yöntemdir. Kategorik veriler için her benzersiz kategoriye bir sayı atanmasını sağlar.

Örneğin, bir veri kümesindeki şehir isimlerini label encoding ile şu şekilde kodlayabiliriz:

- **İstanbul: 0**
- **Ankara: 1**
- **İzmir: 2**

Bu şekilde, **her kategori** (şehir) **bir sayıya** karşılık gelir. Label encoding, genellikle sıralı (ordinal) verilerde kullanılır çünkü sayısal dönüşüm, bir **büyüklük ya da sıralama anlamına** gelebilir.

**Avantajları:** Sıralı veriler için uygundur. (Kategoriler arasında büyülüklük ilişkisi varsa (örneğin, *küçük, orta, büyük*), label encoding bu ilişkiyi korur. Basit ve hızlıdır.

**Dezavantajları:** Anlamsal sıralama yaratabilir: Eğer kategorik veriler sıralı değilse (örneğin, renkler gibi), label encoding yanlış anlamlar çıkarmaya neden olabilir. Model, bu sayısal değerler arasında bir ilişki olduğunu düşünebilir (örneğin, "Mavi" = 0 ve "Kırmızı" = 1, *model mavi ve kırmızı arasında bir büyülüklük ilişkisi olduğunu düşünebilir*).

# TEMEL NLP MODELLERİ

## Label Encoding: One-hot Encoding'e Göre Farkı

Label encoding'de **her kategori bir tamsayı** ile kodlanırken, one-hot encoding **her kategori için ayrı bir sütun** oluşturur ve o sütun içinde o kategoriye aitse "1", değilse "0" değeri atanır. One-hot encoding, sıralı olmayan kategorilerde label encoding'in dezavantajını ortadan kaldırır.

Word2Vec  
GLOVE  
Fast Text  
Skip-Gram  
CBOW  
Sürekli Kelime Torbası

# TEMEL NLP MODELLERİ

**Rare Encoding:** Özellikle veri setlerinde **az görülen (rare) kategorik değerlerin kodlanması** için kullanılan bir yöntemdir. Veri setlerinde bazı kategorik değerler çok nadir görülmüyorsa, bu değerlerin nadirliği **modelin tahmin gücünü** etkileyebilir. Bu durumda nadir değerler birleştirilebilir veya "diğer" kategorisi altında grüplanarak kodlanabilir. Bu yaklaşım, veri setini basitleştirip **modele fazla bilgi yüklemesini** engeller ve **daha dengeli bir analiz** sağlar.

**Örneğin;** Bir veri setinde kullanıcıların ülke bilgisi yer alıyor diyelim. 1000 kullanıcının 990'ı Almanya, Fransa veya İtalya'dan, diğer 10 kişi ise başka ülkelerden geliyorsa, nadir olan bu ülkeler "Diğer" olarak kodlanabilir. Böylece model sadece Almanya, Fransa, İtalya ve "Diğer" olarak dört kategori ile çalışır.

Bu kodlama ile veri setindeki karmaşıklık azaltılır ve modelde **aşırı uyum (overfitting)** riski düşer.

# TEMEL NLP MODELLERİ

Bu veri setinde bir otomobil bayisinin markalara göre sattığı araba sayısı yer alıyor. 10 olarak belirlenen eşik değerin altında kalan markalar rare olarak etiketlenmiş ve diğer kategorisi altına alınmış.

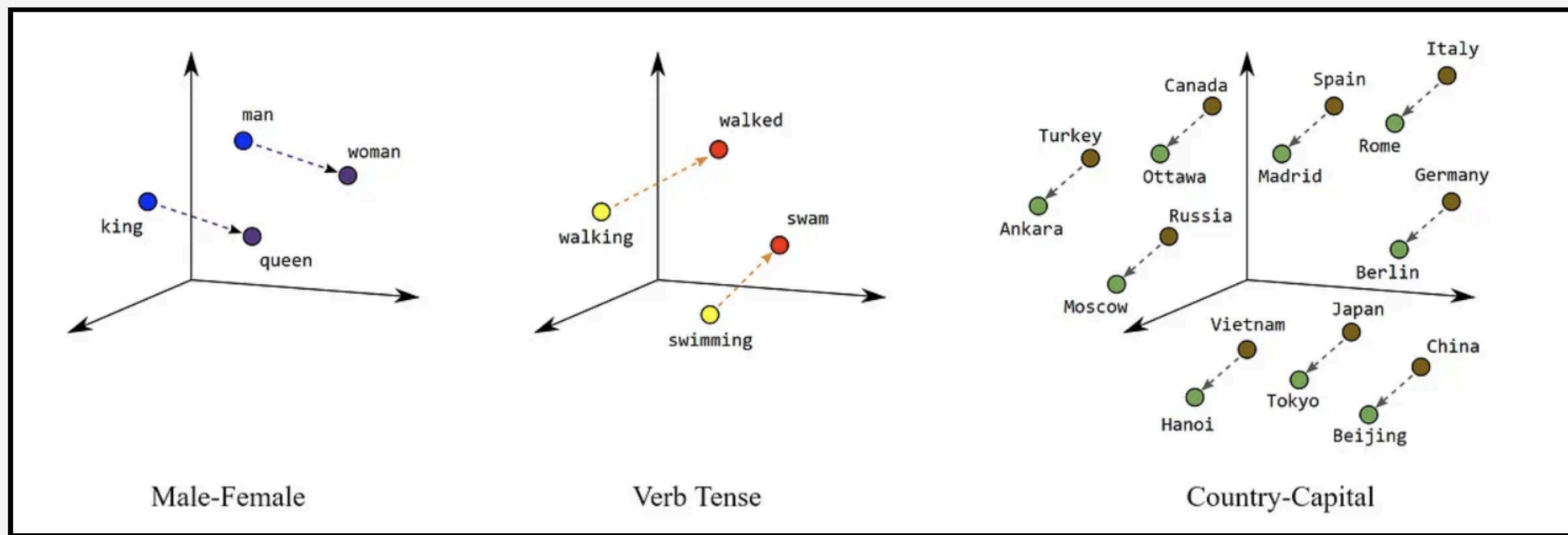
CAR_BRAND	CAR_BRAND_COUNT	CAR_BRAND	CAR_BRAND_COUNT
Chevrolet	82	Chevrolet	82
Audi	53	Audi	53
Rolls Royce	3	Renault	95
Renault	95	Nissan	48
Bentley	2	Bmw	32
Nissan	48	Hyundai	64
Bmw	32	Rare (Rolls Royce, Bentley, Saab)	14
Hyundai	64		
Saab	9		

Tablo 1

# TEMEL NLP MODELLERİ

**EMBEDDING:** Embedding, bir dilin veya verilen verideki kelimelerin tek tek, daha az boyutlu bir uzayda **gerçek değerli vektörler** olarak ifade edilmesidir. Çok detayına girmeden şöyle özetleyebiliriz.

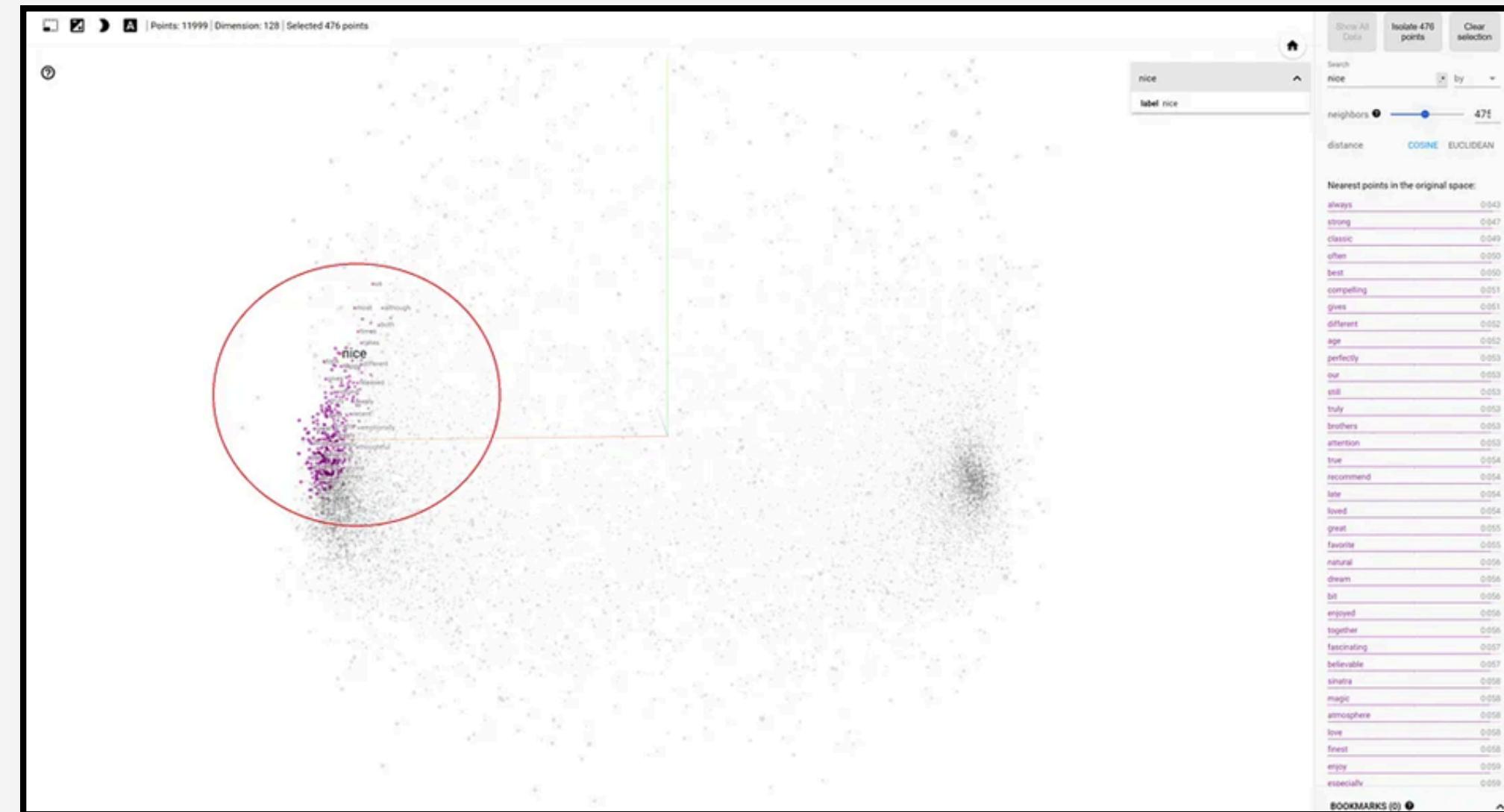
Matriste çok fazla **0 bulunma problemi (sparse matrix)** embedding ile çözülmüştür. Ayrıca, yakın anlamlı olan kelimeler vektör uzayında birbirine yakın olarak bulunur. Örnek verecek olursak:



Görsel 14

# TEMEL NLP MODELLERİ

**Nearest Neighborhood algoritması** ile yakın anlamlı kelimeler birbirine yakın yerleştirilirler.  
**Principal Component Analysis (PCA)** ile de daha düşük bir uzayda ifade edebildiğimiz embedding oluşturabiliriz.



Görsel 15

# TEMEL NLP MODELLERİ

**Word Embeddings(Word2vec, glove, fastText):** Word2Vec, 2013 yılında Google'da Tomas Mikolov tarafından geliştirilen kelime ilişkilendirmesini öğrenmek için kullanılan bir tekniktir.

**Word2vec**, herhangi bir kelimeyi temsil etmek için bir sayı listesi (vektör) kullanır. Vektörler arasındaki **kosinüs benzerliği**, kelimeler arasındaki anlamsal benzerlik seviyesini gösterir. Word2vec'teki algoritmalar bir **sinir ağı modeli kullanır**, böylece bir kez eğitilmiş bir model eş anlamlı ve zıt anlamlı kelimeleri belirleyebilir veya kısmen tamamlanmamış bir cümleyi tamamlamak için bir kelime önerebilir.

**Word2Vec** kullanılarak kelimelerin temsili iki ana yöntemle yapılabilir. Birisi **CBOW**, diğer ise **Skip Gram** yöntemidir.

# TEMEL NLP MODELLERİ

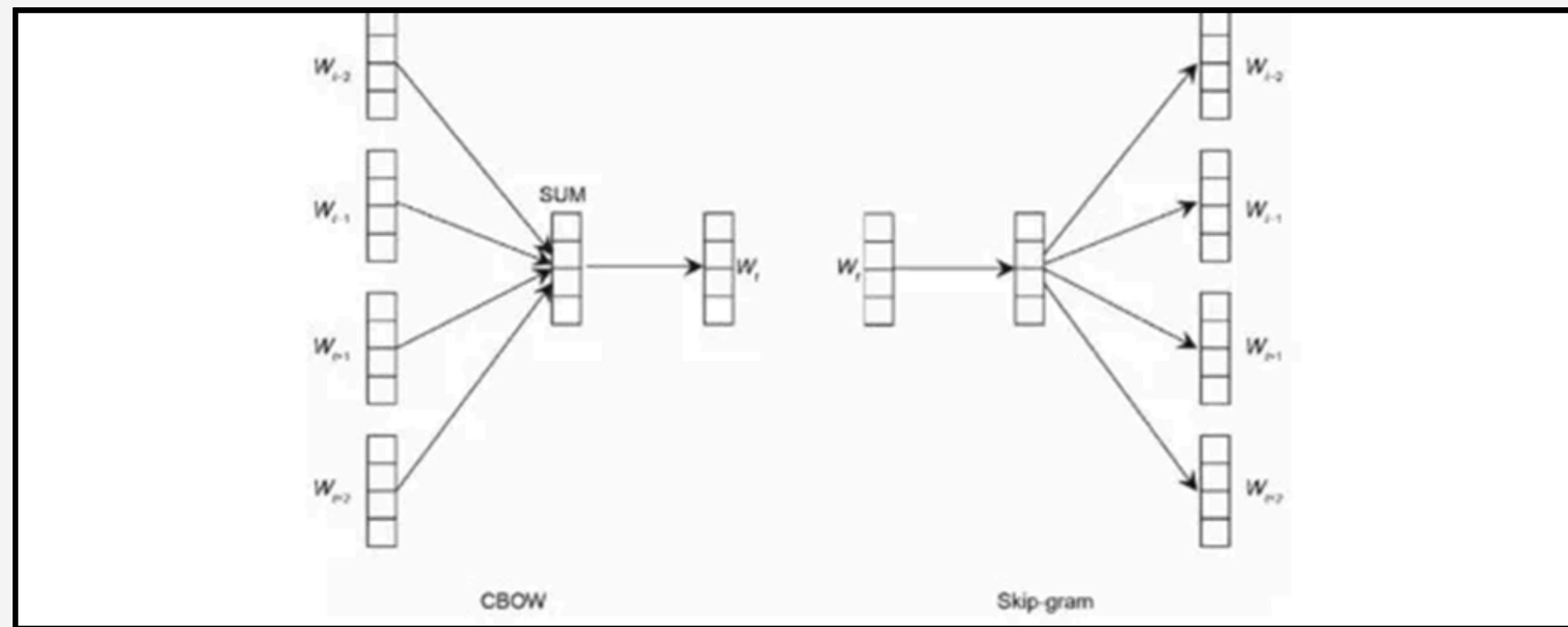
**Sürekli Kelime Torbası (CBOW) Yöntemi:** Bu yöntem, kelimelerin bağlamına dayalı olarak cümlenin ortasına yerleştirilebilecek kelimeleri tahmin ederek eksik bir cümleyi tamamlamaya yardımcı olur. Tahmin bağlamı, tahmin edilen kelimededen önceki ve sonraki birkaç kelimeye bağlıdır. Bu yöntemlere **sözcük torbası yöntemleri** denir çünkü sözcüklerin bağlam içindeki sırası önemli değildir.

# TEMEL NLP MODELLERİ

**Skip-Gram Yöntemi:** Bu yöntem **CBOW** modelinin yaptığından tersini elde etmeye çalışır. Bir **merkez kelimeden yola çıkarak** bağlam kelimelerinin (komşu kelimelerinin) **tahminlerini yapmak** için kullanılır. Burada, bağlam kelimeleri çıktıyken merkez kelime girdidir. Tahmin edilmesi gereken birden fazla bağlam kelimesi olduğundan, bu sorunu zorlaştırır.

# TEMEL NLP MODELLERİ

**CBOW** ve **Skip**: Gram modellerinin temel mimarisi aşağıda gösterilmiştir.



Görsel 16

**CBOW**: merkez sözcüğü tahmin etmek için bir penceredeki bağlam sözcüklerini kullanır. **Skip-gram**: Bir penceredeki bağlam sözcüklerini tahmin etmek için merkez sözcüğü kullanır.  $W_t$ , cümledeki t. kelimedir.

# TEMEL NLP MODELLERİ

**Örneğin**, Skip-gram modelinde “Bugün erken kalkıp kahvaltı hazırladım.” cümlesi için pencere boyutu 5 iken ‘kalkıp’ kelimesi girdi olarak alınır ve ‘bugün’, ‘erken’, ‘kahvaltı’ , ‘hazırladım’ kelimeleri tahmin edilir.

Fakat CBOW modelinde “Bugün erken kalkıp kahvaltı hazırladım.” cümlesinde ‘bugün’, ‘erken’, ‘kahvaltı’ ve ‘hazırladım’ kelimeleri girdi olarak alınır. Amaç ise bu **kelimelerin arasına gelecek en uygun kelimeyi bulmaktır**.

	Bugün	erken	kalkıp	kahvaltı	hazırladım
Bugün	0	1	0	0	0
erken	1	0	1	0	0
kalkıp	0	1	0	1	0
kahvaltı	0	0	1	0	1
hazırladım	0	0	0	1	0

Görsel 17

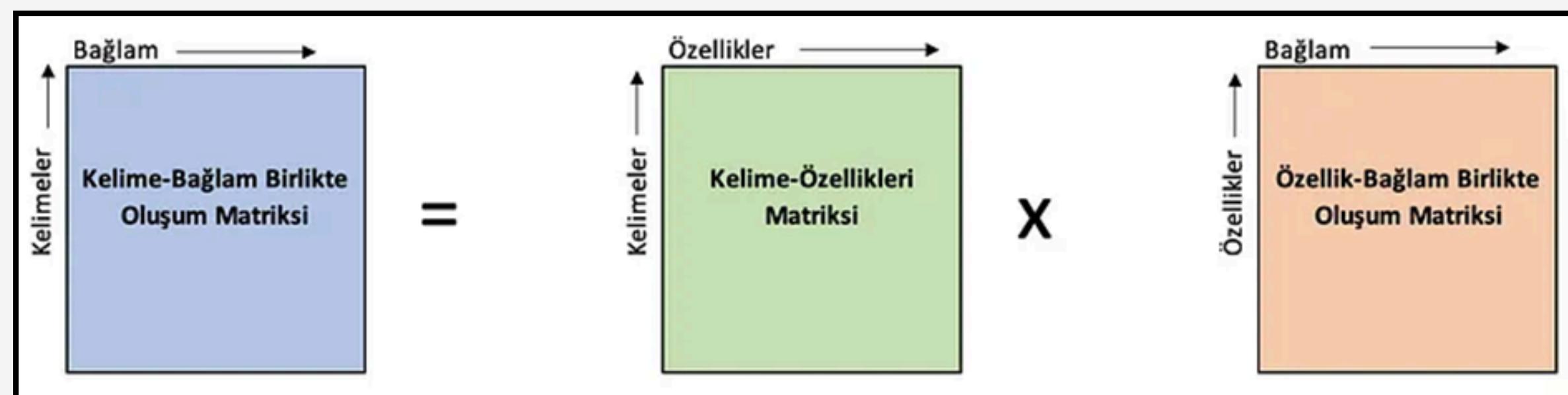
# TEMEL NLP MODELLERİ

**GLOVE:** GloVe bir diğer **kelime gömme yöntemidir**. Ancak, gömme matrisini oluşturmak için farklı denklemler ve mekanizma kullanır. Stanford'da açık kaynaklı bir proje olarak geliştirilen ve 2014'te piyasaya sürülen bu model, **denetimsiz bir öğrenme algoritması** (*Denetimsiz öğrenme, makine öğrenmesinde verilerin etiketlenmediği durumlarda kullanılan bir öğrenme teknigidir.*) kullanır.

**Denetimsiz algoritmalar**, verilerin **istatistiklerine dayanır**. Skip-gram, CBOW gibi modeller birlikte oluşum matrisini (co-occurrence matrix) kullanmaz yani **kelimeler arasındaki anlamsal ilişkileri yakalayamaz**. Fakat GloVe modeli olasılık istatistiklerini kullanarak bu sorunu çözmeyi amaçlamaktadır.

# TEMEL NLP MODELLERİ

**Birlikte oluşum matrisini** oluşturuktan sonra, GloVe modeli bu matrisi **çarpanlara ayırarak** daha düşük boyutlu (kelime x özellikleri) bir matris verir; burada her satır karşılık gelen kelime için bir vektör temsili verir. Genel olarak, bu bir “**yeniden yapılanma kaybını**” en aza indirerek yapılır. Bu kayıp, yüksek boyutlu verilerdeki **varyansın** çoğunu açıklayabilen **daha düşük boyutlu temsilleri bulmaya çalışır.**



Görsel 18

# TEMEL NLP MODELLERİ

## Word2vec ve Glove Farkı Nedir?

Kelime gömme tekniğinde **her kelime bir vektöre eşlenir** ve vektör değerleri, sinir ağları veya birlikte oluşum matrisleri gibi bazı modellerle hesaplanır. Glove modeli, tüm kelime topluluğundan küresel bağlamda yararlanarak **kelimelerin birlikte bulunma sıklıklarına dayanır**. Öte yandan **Word2vec**, metinleri bir sinir ağı için **eğitim verileri olarak alır**. Ortaya çıkan vektörler, kelimelerin benzer bağlamlarda görünüp görünmediğini yakalar. Kısaca bu iki model, eğitilme biçimleri bakımından farklılık gösterir ve bu nedenle son derece farklı özelliklere sahip kelime vektörleri üretir.

Ancak uygulamada, bu modellerin her ikisi de birçok görev için benzer sonuçlar verir. Bu modellerin üzerinde eğitildiği veri seti, vektörlerin uzunluğu gibi faktörlerin, modellerin kendisinden daha büyük bir etkisi olduğu görülmektedir. *Örneğin, bir tıbbi uygulamanın özelliklerini türetmek için bu modelleri kullanıyorsak, tıbbi alandaki veri kümlesi üzerinde eğitim alarak performansı önemli ölçüde artırabiliriz.*

# TEMEL NLP MODELLERİ

## Word2vec ve Glove Farkı Nedir?

### 1. Eğitim Yöntemi

- Word2vec: **Sinir ağı tabanlı** bir modeldir ve kelimeleri **yerel bağamlara** dayanarak öğrenir. İki temel yaklaşımı vardır: Continuous Bag of Words (CBOW) ve Skip-gram.
- GloVe: **Küresel bağlam** bilgisi kullanarak, **kelimelerin birlikte bulunma sıklıklarına** dayanan bir modeldir. Kelimelerin **tüm metin üzerindeki istatistiklerini** değerlendirir.

### 2. Model Yapısı

- Word2vec: Yerel bağamlar üzerinden öğrenme yaptığı için daha hızlıdır ve genellikle daha düşük hesaplama kaynakları gerektir.
- GloVe: Kelimelerin tüm metin üzerindeki ilişkilerini dikkate alır, bu nedenle daha karmaşık bir modeldir ve **daha fazla bellek** kullanabilir.

### 3. Vektör Temsili:

- Word2vec: Kelimelerin **benzer bağamlarda** görünüp görünmediğini yakalayan **vektörler** üretir.
- GloVe: Kelimelerin **global istatistiklerine** dayalı olarak, **kelimeler arasındaki benzerlikleri ve ilişkileri** daha iyi temsil eden vektörler oluşturur.

# TEMEL NLP MODELLERİ

## Word2vec ve Glove Farkı Nedir?

### 4. Hız ve Verimlilik:

- **Word2vec**: Genellikle **daha hızlı eğitim** süresine sahiptir.
- **GloVe**: Daha yavaş olabilir, çünkü **daha fazla hesaplama ve veri analizi** gerektirir.

### 5. Uygulama Alanları:

- Her iki model de benzer sonuçlar verir, ancak belirli görevlerde bir model diğerine göre üstünlük gösterebilir.

*Örneğin, belirli bir alan için (tıbbi gibi) eğitilmiş bir model, o alandaki uygulamalarda daha iyi performans gösterebilir.*

### 6. Veri Seti Etkisi:

- Her iki model de, üzerinde eğitim aldıkları **veri setinin kalitesi ve büyüklüğünden büyük ölçüde etkilenir.**

# TEMEL NLP MODELLERİ

## Fast Text

- FastText, **Facebook AI Research** tarafından geliştirilen bir kelime temsil modelidir. Word2Vec'e benzer şekilde kelime vektörleri oluşturur, ancak önemli bir farkı vardır: FastText, kelimeleri **karakter n-gramları kullanarak temsil eder**. Bu da modelin nadir kelimeleri ve farklı dil yapılarını daha iyi öğrenmesine yardımcı olur. Yani, kelimelerin iç yapısını (alt kelimeleri ve ekleri) dikkate alarak daha etkili bir vektör temsil sunar.
- Bu özellik, dil bilgisel yapıları **daha iyi yakaladığı ve nadir kelimeleri daha iyi öğrendiği için** Word2Vec'den farklıdır ve bazı görevlerde daha etkilidir.

# TEMEL NLP MODELLERİ

## FastText'in Avantajları

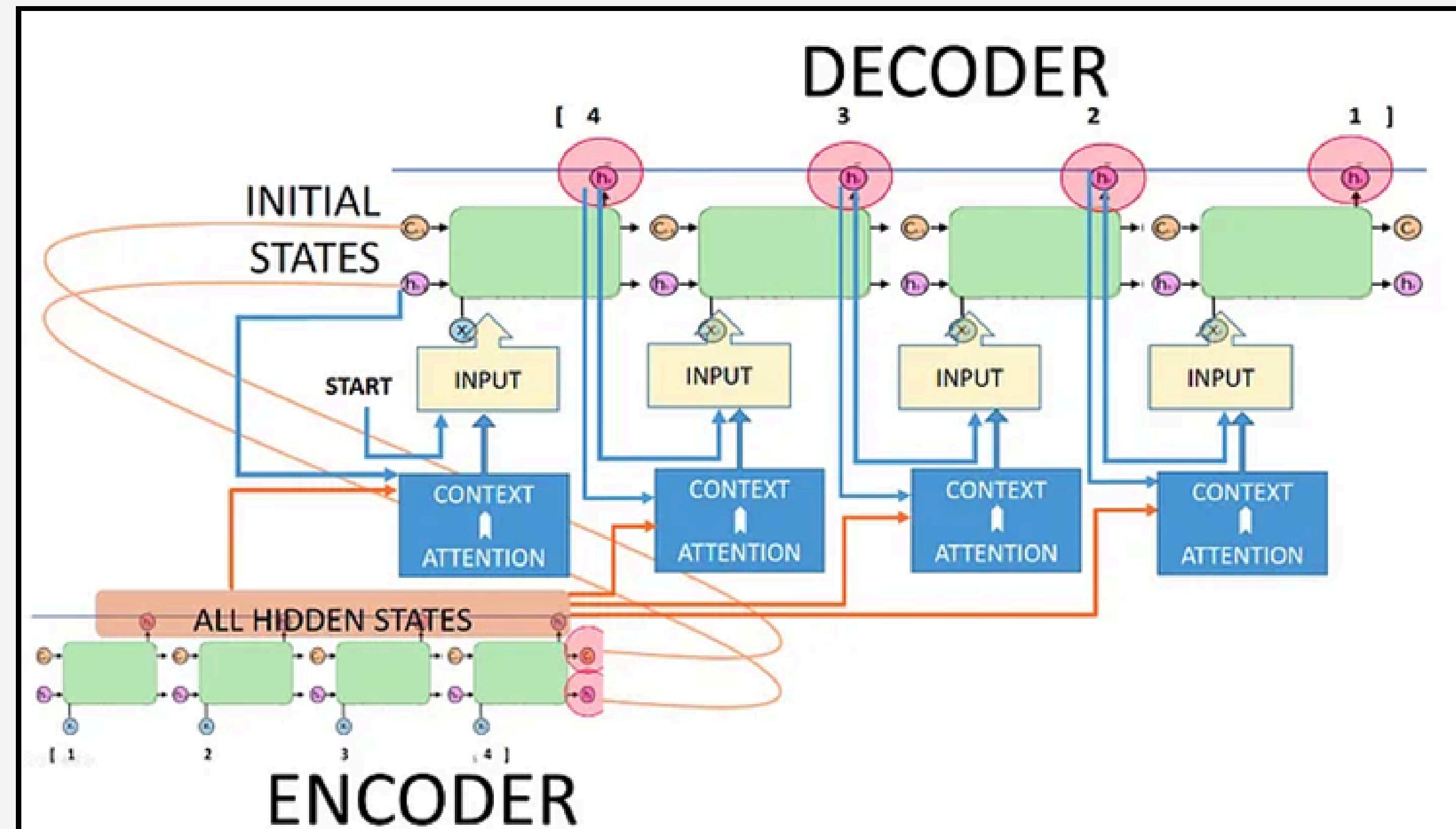
- **1. Hızlı Eğitim ve Tahminler:** Büyük veri kümelerini hızlı bir şekilde işleyebilir ve tahminler yapabilir. Bu, **büyük ölçekli metin sınıflandırma** projeleri için önemlidir.
- **2. Kelime Gömme (Word Embedding):** FastText, metindeki her kelimenin **anlamsal temsilini** yakalar. Bu, kelimenin anlamını ve benzerliklerini öğrenme yeteneği sağlar. Özellikle **kelime seviyesinde dil analizi** gerektiren görevler için idealdir.
- **3. Çoklu Dil Desteği:** FastText, birçok farklı dilde çalışabilir ve bu diller için **özel kelime gömme (word embedding) matrisleri** oluşturabilir. 157 farklı dilde *Common Crawl* ve *Wikipedia* içerikleri ile eğitilmiş hazır modelleri mevcuttur.

# ATTENTION MEKANİZMASI

Attention mekanizması, bir modelin girdi verisinin belirli bölümlerine “**daha fazla dikkat etmesi**” gerektiğini belirlemesine olanak tanır. Bu kavram, insanların yeni bilgileri işlerken, mevcut bilgileri ve bağamları göz önünde bulundurarak **odaklanma yeteneğinden esinlenmiştir**. NLP’de, bir cümle veya metin parçası işlenirken, bazı kelimelerin veya ifadelerin anlamı üzerinde daha fazla durulması gerekebilir. Attention mekanizması, bu önemli bölgelere daha fazla “dikkat etmek” ve modelin performansını artırmak için kullanılır.

# ATTENTION MEKANİZMASI

Attention Mekanizması Nasıl Çalışır?



Görsel 19

# ATTENTION MEKANİZMASI

## Context Vektörü Nasıl Hesaplanır?

hs : Encoder tarafından üretilen tüm ht'ler

ht : Bir önceki adımda decoder tarafından üretilen tüm ht

W : ağırlıklar

$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s) = \begin{cases} \mathbf{h}_t^\top \mathbf{W} \bar{\mathbf{h}}_s & [\text{Luong's multiplicative style}] \\ \mathbf{v}_a^\top \tanh(\mathbf{W}_1 \mathbf{h}_t + \mathbf{W}_2 \bar{\mathbf{h}}_s) & [\text{Bahdanau's additive style}] \end{cases} \quad (4)$$

tanh : aktivasyon fonksiyonu ()

v : tek katmanlı model

Dolayısıyla burada derin öğrenme modeli, kendi hesaplamalarıyla (feedforward ve back propagation) en uygun W,tanh,v değerlerine ulaşacaktır.

# ATTENTION MEKANİZMASI

## Context Vektörü Nasıl Hesaplanır?

Attention ağırlıklarının bulunması için; buradaki ağırlıklar her bir decoder gizli katmanı için olan ağırlıklardır. Toplamanın 1 olacağından dolayı olasılık dağılımının hesaplamak için sigmoid fonksiyonu kullanılmıştır.

$$\alpha_{ts} = \frac{\exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s))}{\sum_{s'=1}^S \exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_{s'}))}$$

[Attention weights]

Context vektor hesaplanması için de her bir katman ile bulunan attention ağırlığını çarpıp topluyoruz. Yani burada şunu söylüyoruz. Hesapladığımız attention ağırlığı yüksek ise, gelen hidden state'in etkisi büyük olacak, attention ağırlığı düşük ise, gelen hidden state'in etkisi küçük olacaktır.

$$\mathbf{c}_t = \sum_s \alpha_{ts} \bar{\mathbf{h}}_s$$

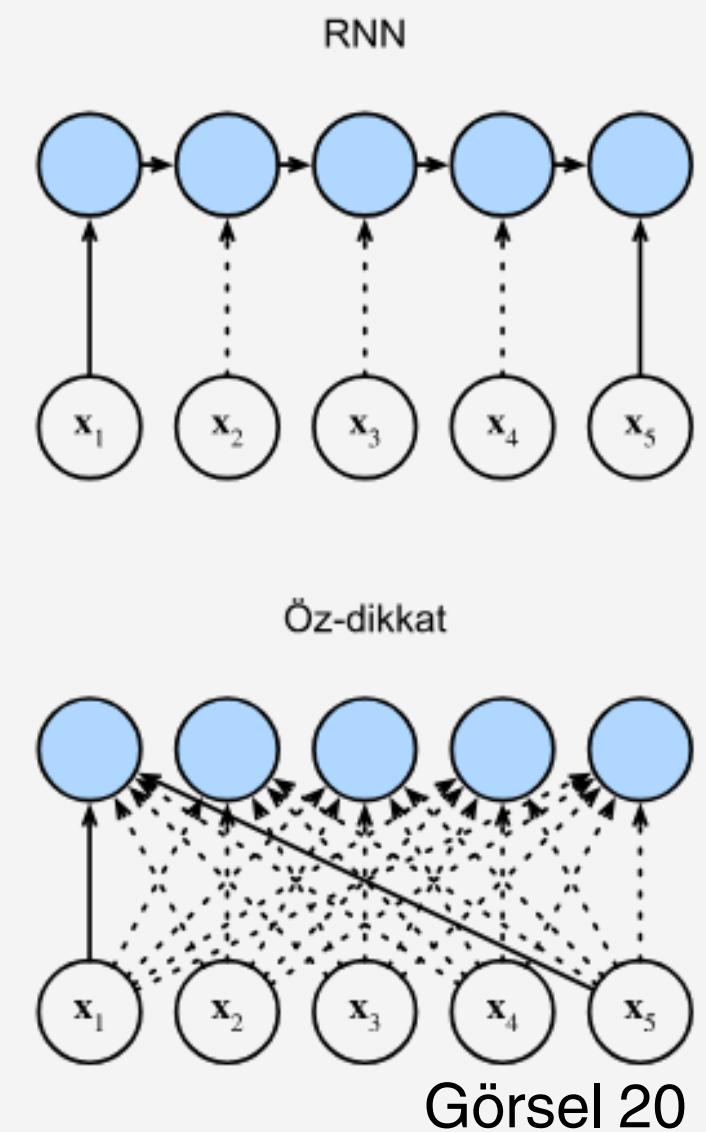
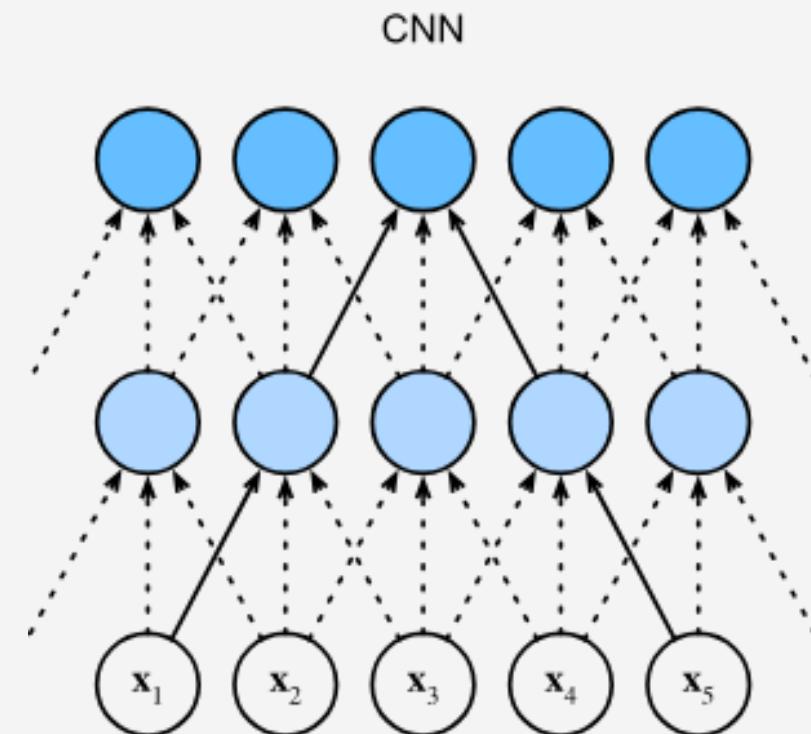
[Context vector]

# ATTENTION MEKANİZMASI

## Attention Mekanizmasının Çeşitleri

### 1. Self-Attention (Kendi Üzerine Dikkat - Öz Dikkat)

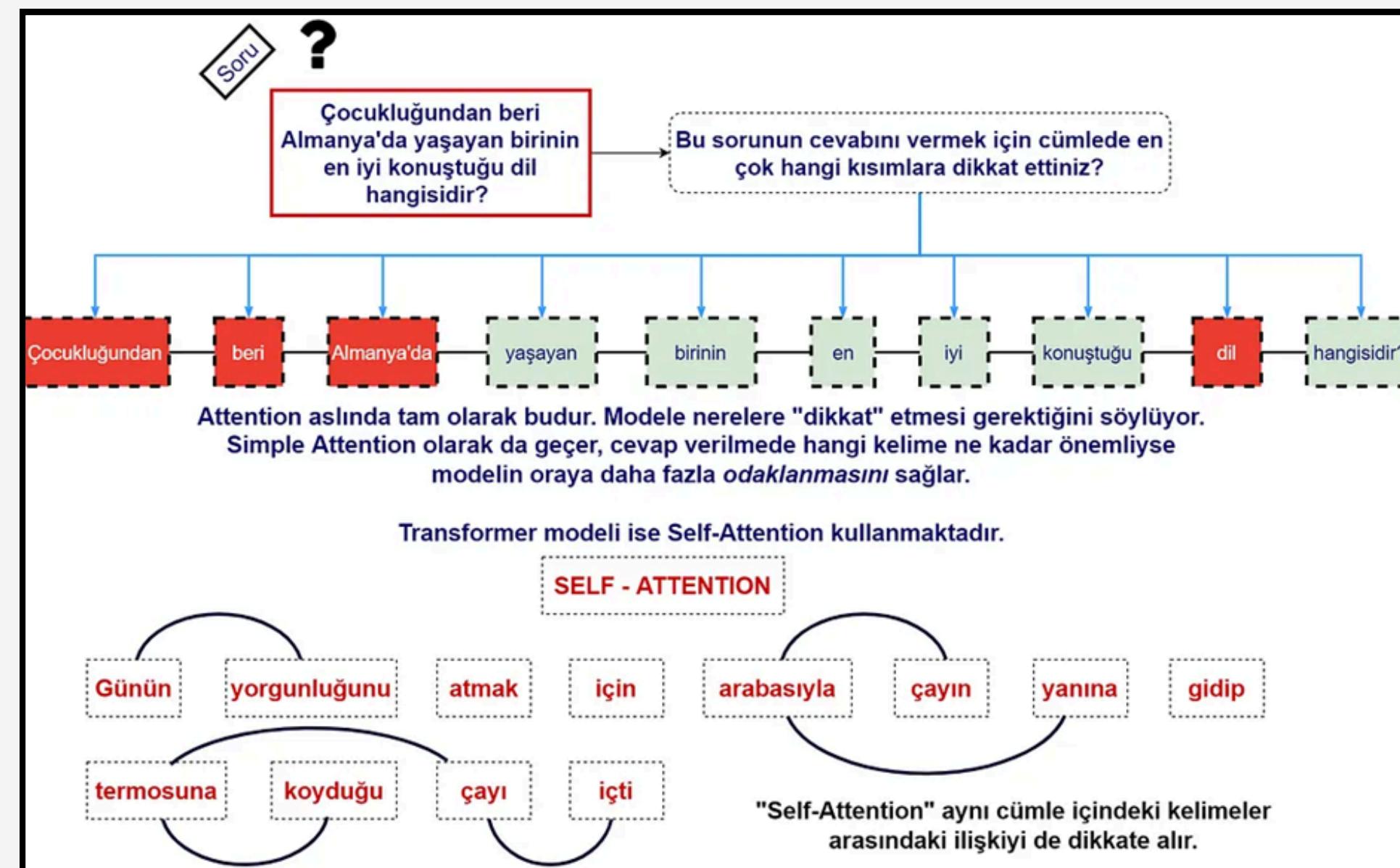
Self-attention, bir cümlenin veya metnin içindeki her elemanın diğer tüm elemanlarla olan ilişkisini değerlendirir. Bu mekanizma, modelin metin içindeki bağlamsal ve anlamsal ilişkileri öğrenmesine olanak tanır. Örneğin, "The animal didn't cross the street because it was too tired," cümlesinde "it" kelimesinin "animal" kelimesine referans verdiğini anlamak self-attention sayesinde mümkün olur. Model, her kelimenin diğer kelimelerle nasıl bir bağ kurduğunu anlamaya çalışır.



# ATTENTION MEKANİZMASI

## Attention Mekanizmasının Çeşitleri

### 1. Self-Attention (Kendi Üzerine Dikkat - Öz Dikkat)



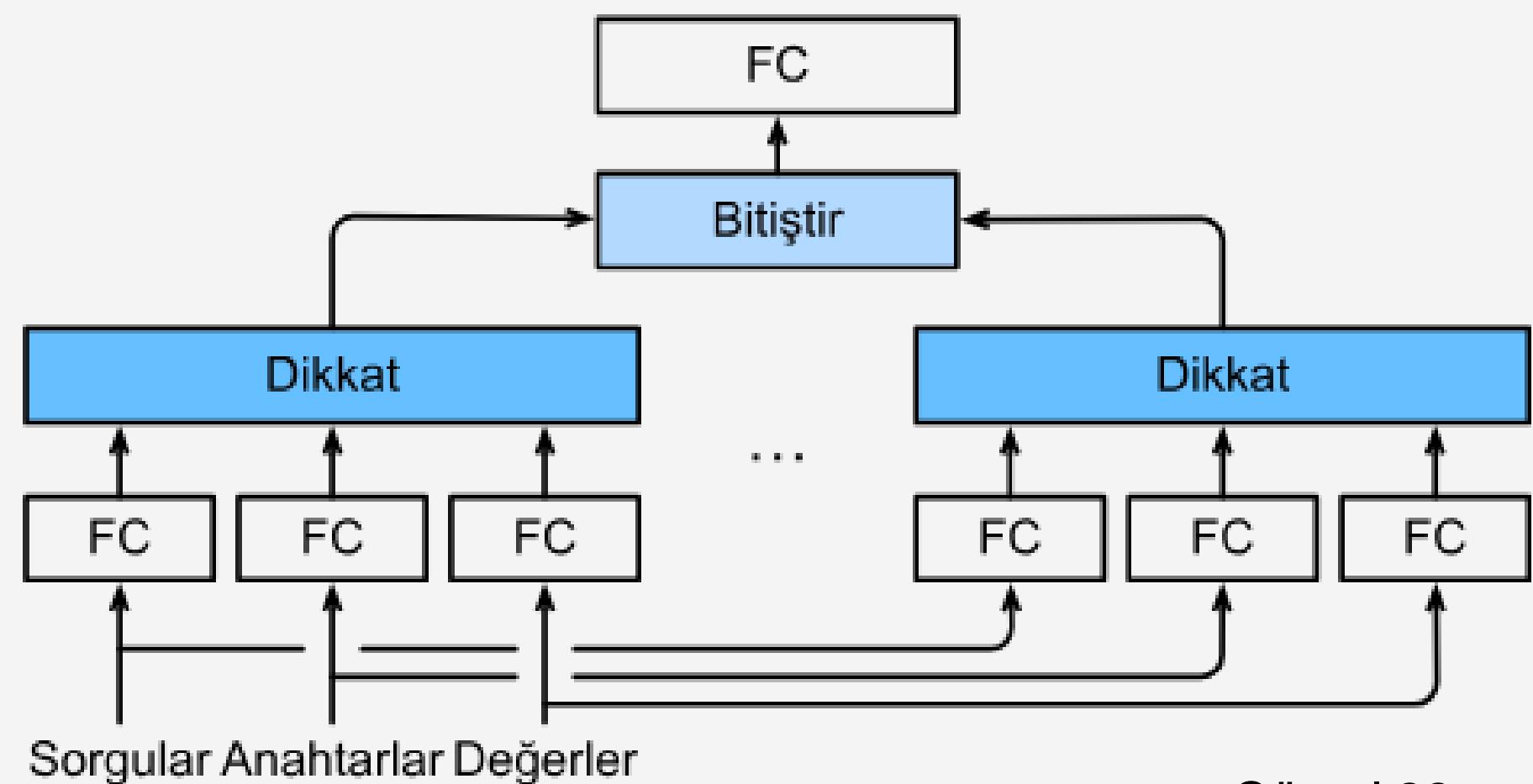
Görsel 21

# ATTENTION MEKANİZMASI

## Attention Mekanizmasının Çeşitleri

### 2. Multi-Head Attention (Çok Başlıklı Dikkat)

Multi-head attention, self-attention mekanizmasının bir genişlemesidir ve veriyi parallel olarak işleyebilme yeteneğine sahiptir. Model, birden fazla "başlık" (head) kullanarak verinin farklı bölmelerine aynı anda odaklanır. Her başlık, veriyi farklı bir açıdan işler; böylece model birden fazla bağlamı ve ilişkiyi eş zamanlı olarak öğrenebilir. Bu özellik, modelin daha zengin ve derin anlamları yakalayarak genel anlamaya yeteneğini artırır.



Görsel 22

# ATTENTION MEKANİZMASI

## 3. Cross-Attention (Çapraz Dikkat)

Cross-attention, özellikle encoder-decoder yapılarında kullanılır. Decoder, encoder tarafından üretilen temsillere dikkat eder ve hedef çıktıyı oluştururken bu bilgilere dayanarak karar verir. Örneğin, bir dil çevirisinde, decoder kaynak cümledeki belirli bir kelimeye dikkat ederek hedef cümledeki doğru karşılığı bulur. Bu mekanizma, özellikle metin çevirisi gibi görevlerde kaynak ve hedef metin arasında bağlantı kurmak için kullanılır.

# ATTENTION MEKANİZMASI

## 4. Global Attention (Küresel Dikkat):

Global attention, modelin tüm girdi dizisine dikkat etmesini sağlar. Bu tür dikkat, özellikle metnin tamamını işlemeyi gerektiren özetleme gibi görevlerde kullanılır. Model, cümlenin veya metnin tüm bölümlerine aynı anda odaklanarak genel bağlamı yakalamaya çalışır.

# ATTENTION MEKANİZMASI

## 5. Local Attention (Yerel Dikkat)

Local attention, modelin sadece sınırlı bir girdi penceresine dikkat etmesini sağlar. Özellikle uzun dizilerde hesaplama maliyetini düşürmek ve daha kısa mesafeli bağımlılıkları öğrenmek için kullanılır. Bu dikkat türü, uzun metinlerin işlem süresini optimize ederken önemli ilişkilerin kaybolmasını engeller.

# ATTENTION MEKANİZMASI

## Flash Attention (Hızlı ve Bellek-Etkin Dikkat Mekanizması)

FlashAttention, dikkat hesaplamalarını hızlandırmak ve belleği daha verimli kullanmak için geliştirilmiş bir yöntemdir ve **self-attention** üzerinde çalışır.

Modern derin öğrenme modellerinde, özellikle büyük dil modelleri (LLM'ler) ve dikkat mekanizmaları için kullanılan bir optimizasyon tekniğidir. 2022 yılında "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness" başlıklı makale ile tanıtılan bu teknik, dikkat (attention) işlemlerinin **daha hızlı ve bellek verimli** bir şekilde hesaplanması sağlar.

# ATTENTION MEKANİZMASI

## Flash Attention Nedir ve Neden Gereklidir?

Flash Attention, geleneksel self-attention (kendi üzerine dikkat) mekanizmalarına kıyasla daha hızlı ve bellek açısından daha verimli çalışır. Özellikle büyük modellerde (BERT, GPT-4 gibi) eğitim süreçlerini hızlandırmak ve daha az bellekle daha büyük veri setlerini işleyebilmek amacıyla geliştirilmiştir. Normal self-attention'da her kelimenin (veya token'in) diğer tüm kelimelerle olan ilişkisini öğrenmek için büyük matris çarpımları yapılır. Bu süreç, bellekte büyük bir yük oluşturur ve özellikle uzun dizilerde oldukça zaman alır. Flash Attention, bu süreci optimize ederek hız ve bellek verimliliğini artırır.

# ATTENTION MEKANİZMASI

## Flash Attention'ın Öne Çıkan Özellikleri

**Hız:** Flash Attention, geleneksel self-attention'a kıyasla çok daha hızlı çalışır. Örneğin, BERT-large gibi büyük modellerin eğitim sürelerini üç kata kadar hızlandırır. Bu hız artışı, modelin doğruluğundan ödün vermeden elde edilir, yani performans kaybı yaşanmaz.

**Bellek Verimliliği:** Flash Attention, belleği geleneksel dikkat mekanizmalarına göre çok daha etkin kullanır. Geleneksel self-attention'da bellek kullanımı karekök düzeyinde bir artış göstererek  $O(n^2)$  kompleksiteye sahipken, FlashAttention, bellek ve hesaplama maliyetini optimize ederek  $O(n)$  karmaşıklıkla lineer bir şekilde ölçeklenir. Bu sayede, büyük ölçekli veri setlerinde bellek verimliliği önemli ölçüde artırılmış olur. Bu, FlashAttention'ın daha az bellekle çok daha fazla veri işleyebilecegi anlamına gelir.

# ATTENTION MEKANİZMASI

## Flash Attention'ın Öne Çıkan Özellikleri

**IO-Farkındalığı (Input/Output - Girdi/Cıktı):** Flash Attention, giriş-çıkış işlemlerini (IO) daha etkin yönetir. Bu, dikkat işlemlerinde giriş ve çıkışların nasıl ele alındığını optimize ederek, işlem süresinin kısalmasını sağlar. Girdi ve çıktı verilerinin bellek içindeki yerleşimi daha verimli hale getirilir, böylece modelin belleğe olan bağımlılığı azaltılır.

**Tam Dikkat Mekanizması:** Flash Attention, geleneksel dikkat mekanizmasının eksiksiz bir versiyonudur. Yani, herhangi bir kısayol veya yaklaşık hesaplama yapmaz; bu nedenle, Flash Attention'ın sonuçları geleneksel dikkat mekanizmasının sonuçlarıyla aynıdır. Ancak bu işlemi çok daha hızlı ve verimli yapar.

# ATTENTION MEKANİZMASI

## Flash Attention Nasıl Çalışır?

Flash Attention, dikkat mekanizmasının hesaplanması optimize etmek için iki ana teknik kullanır:

**Tiling (Bloklama):** Dikkat matrislerini küçük bloklara böler ve bu küçük blokları sırayla işler. Bu sayede bellek kullanımını optimize eder ve hesaplama verimliliğini artırır. Geleneksel self-attention'da tüm dikkat matrisi aynı anda hesaplanırken, Flash Attention, bu matrisi küçük parçalara ayırarak işlemi hızlandırır.

**Recomputation (Yeniden Hesaplama):** Geri yayılım aşamasında, dikkat matrislerini yeniden hesaplar ve bu işlem için fazla bellek depolama ihtiyacını ortadan kaldırır. Normalde dikkat skorları bellekte saklanırken, Flash Attention bu skorları gerektiğinde yeniden hesaplar, böylece bellek tasarrufu sağlar.

# ATTENTION MEKANİZMASI

## IO-Farkındalığı Nedir?

Flash Attention'ın en önemli yeniliklerinden biri, IO-Farkındalığıdır. Bu özellik, Flash Attention'ın donanımın bellek hiyerarşisini kullanarak işlem yapmasına olanak tanır. Geleneksel dikkat mekanizmaları, bellek erişimi gecikmelerini göz ardı edebilirken, Flash Attention, bu sorunu IO-Farkındalığı ile çözer. Özellikle GPU'lar gibi hızlandırıcıların bellek erişim hızları ve kapasiteleri arasında uyum sağlamak için geliştirilen bu yaklaşım, bellek erişimini optimize eder ve daha hızlı hesaplamalar yapılmasını sağlar.

IO-Farkındalığı, aynı zamanda donanımın bir "kara kutu" (black box) gibi değil, bir kaynak olarak görülmemesini sağlar. "Kara kutu" terimi, sistemin nasıl çalıştığını dair bilgimizin sınırlı olduğu durumları ifade eder. Flash Attention, bu belirsizlikleri ortadan kaldırarak donanımın özelliklerini en iyi şekilde kullanır.

# TRANSFORMER MODELLERİ

- İlk kez Google'in 2017'deki "**Attention is All You Need**" makalesinde tanıtılan transformerlar, yapay zekada devrim yaratmıştır.
- Özellikle doğal dil işleme gibi alanlarda hız ve verimlilik sağlamaları sayesinde, **makine öğreniminindeki en güçlü ve yaygın** modellerden biri haline gelmiştir.
- Transformer modelleri, sıralı verilerdeki (örneğin bir cümledeki kelimeler gibi) ilişkileri izleyerek bağlamı ve anlamı öğrenen sinir ağlarıdır. Bu modeller, uzak veri öğeleri arasındaki ince bağlantıları tespit etmek için dikkat mekanizmasını (attention) kullanır.

# TRANSFORMER MODELLERİ

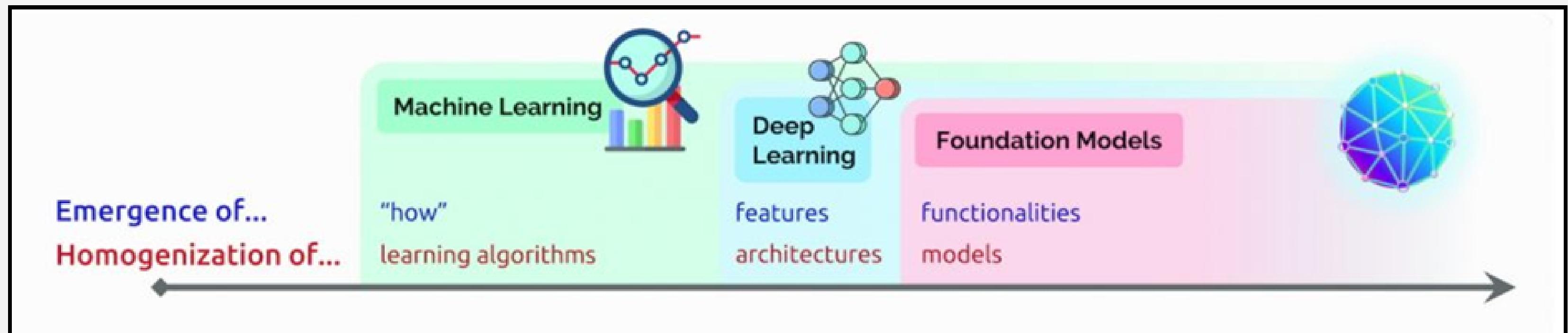
- Geleneksel **RNN** (Recurrent Neural Network - Tekrarlayan Sinir Ağı) ve **LSTM** (Long Short-Term Memory - Uzun Kısa Süreli Bellek) gibi önceki modellere kıyasla, transformer modeller **daha hızlı** ve **daha etkili** şekilde eğitilebilirler. Bunun ana sebebi, **paralel işleme** kabiliyetine sahip olmaları ve "**attention mekanizması**"nı kullanarak verinin farklı bölümleri arasındaki bağlantıları daha iyi anlamalarıdır.
- Stanford araştırmacıları, bu modellerin etkisini vurgulayarak 2021'de "**temel modeller**" olarak adlandırmış ve yapay zekada yeni bir dönemin kapısını açtığını belirtmiştir.
- Transformer modelleri, geniş ölçekleri ve kapsamlarıyla yapay zeka dünyasında çığır açmış, birçok ileri düzey uygulamada kullanılmaktadır.

# TRANSFORMER MODELLERİ

## Transformer İsminin Ortaya Çıkışı

- Transformer isminin ortaya çıkışı, aslında modelin en önemli bileşenlerinden biri olan "**attention**" (ilgi) mekanizmasına dayanıyor.
- Başlangıçta modeli "**İlgi Ağı (Attention Network)**" olarak adlandırmayı düşünen araştırmacılar, bu ismin yeterince güçlü olmadığını hissettiler.
- Sonrasında, ekibin bir üyesi Jakob Uszkoreit, modelin temsilleri dönüştürme (transform) yeteneğini öne çikaran Transformer ismini önerdi.
- Bu isim, modelin veri temsillerini dönüştüren yapısına uygun bir kelime oyunu olarak tercih edildi.

# TRANSFORMER MODELLERİ



Görsel 23

# TRANSFORMER MODELLERİ

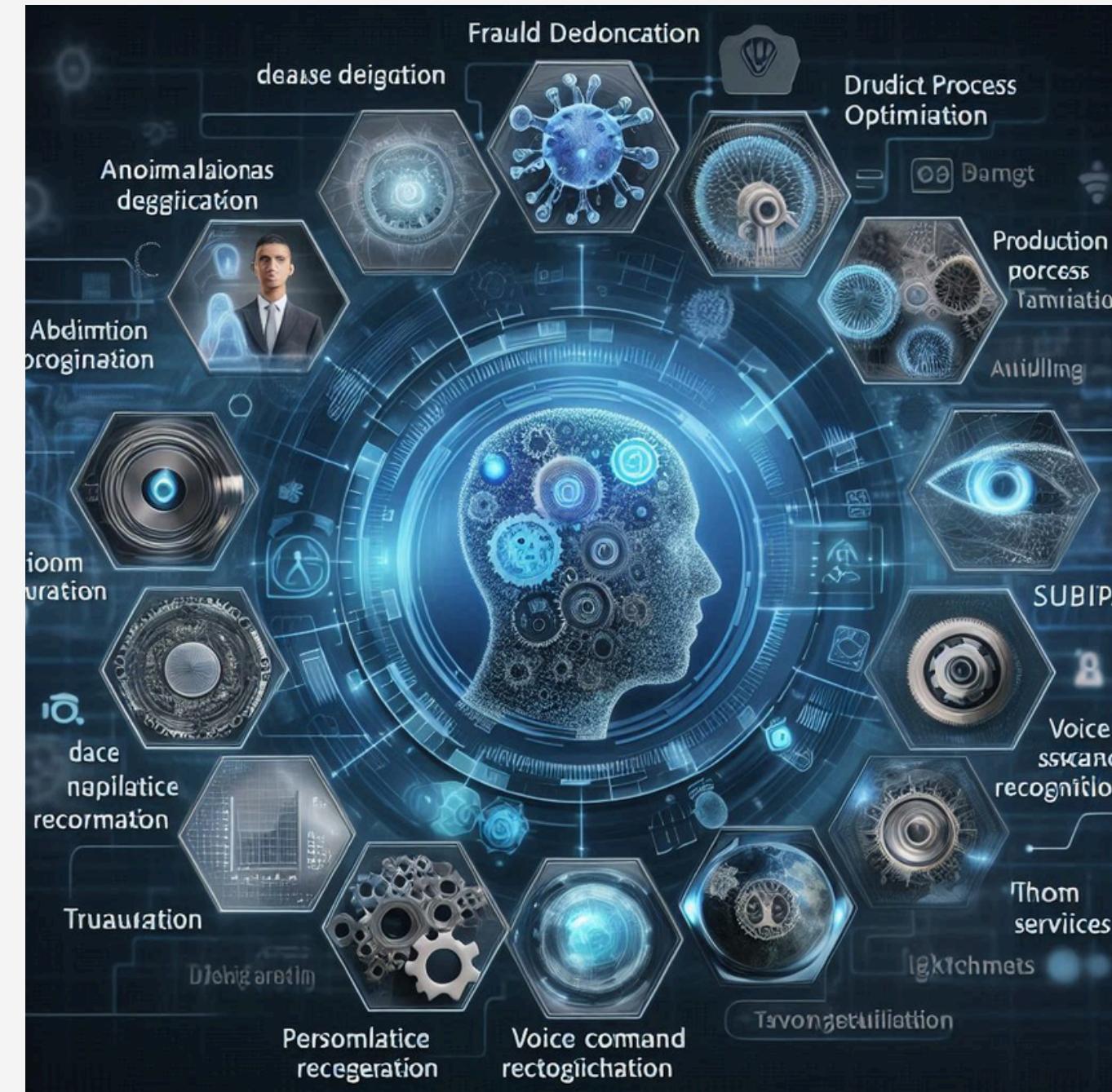
## En Popüler Transformer Modelleri (2024-KASIM)

- GPT-4 (OpenAI)
- BERT (Google)
- T5 (Google)
- ChatGPT (OpenAI)
- LLAMA 2 (Meta)
- Claude (Anthropic)

# TRANSFORMER MODELLERİ

## Kullanım Alanları

- Anomali tespiti
- Hastalık teşhisi
- Dolandırıcılık önleme
- Üretim süreci optimizasyonu
- Kişiselleştirilmiş öneriler
- **Doğal dil işleme**
- Görüntü tanıma
- Sesli komut algılama
- Çeviri hizmetleri
- Sesli asistan geliştirme



# TRANSFORMER MODELLERİ

## Daha Az Etiket, Daha Fazla Performans

- NVIDIA'nın kurucusu ve CEO'su Jensen Huang GTC'de (GPU Teknolojisi Konferansı) yaptığı açılış konuşmasında, "Transformerlar kendi kendine denetimli öğrenmeyi mümkün kıladı ve yapay zeka alanında ışık hızında bir atılıma sebep oldu" dedi.
- Burada "**Transformerlar kendi kendine denetimli öğrenmeyi mümkün kıladı**" ifadesi, transformerların etiketli veri ihtiyacını azaltarak, etiketsiz veya az etiketli verilerle çalışabilmesini sağladığını işaret eder.
- Geleneksel sinir ağları, etiketlenmiş veri gerektirirken, transformerlar matematiksel dikkat mekanizması sayesinde bu gereksinimi ortadan kaldırır.
- Bu, trilyonlarca veri üzerinde etkili bir şekilde çalışmasına olanak tanır ve yapay zeka alanında daha hızlı ilerlemeler kaydedilmesini sağlar. **Paralel işleme avantajı** da bu süreci hızlandırır ve performansını yükseltir.

# TRANSFORMER MODELLERİ

## Transformer Modelinin Temel Bileşenleri

Transformer modelleri, temel olarak iki ana bileşenden oluşur: Encoder ve Decoder. Her biri, birden fazla katmandan oluşan bu bileşenler, farklı ama birbiriyle ilişkili görevleri yerine getirirler.

**Encoder:** Metni alır ve her bir kelime veya token için bir temsil (representation) oluşturur. Bu temsiller, metnin anlamını ve kelimeler arasındaki ilişkileri içerir. Encoder, **metnin anlamını bir dizi vektör olarak kodlar**, bu da decoder tarafından kullanılabilir.

**Attention Mekanizması:** Hem encoder hem de decoder içinde bulunan, modelin metnin hangi bölümlerine “dikkat etmesi” gerektiğini belirleyen bir özel mekanizmadır. Bu sayede model, özellikle çeviri gibi görevlerde, metnin belirli bölümleri arasındaki bağlantıları daha etkin bir şekilde kurabilir.

**Decoder:** Encoder tarafından oluşturulan temsilleri kullanarak, hedef dili veya metni üretir. Decoder, bir çıktı üretmek için hem encoder'dan gelen bilgileri hem de kendi ürettiği önceki çıktıları kullanır.

# ZAMAN SERİSİ VE SIRALI VERİ

- Sıralı veriler, doğal dil işleme alanında kritik bir öneme sahiptir çünkü her kelimenin anlamı, sırasındaki diğer kelimelelere bağlıdır.
- Bu bağlamda, dil verileri zaman serisi verileri gibi işlenmelidir.
- Kelimelerin ardışık konumları, cümlenin genel bağlamını belirler ve doğru bir anlam çıkarmak için sıralı yapıların analiz edilmesi gereklidir.
- Bu nedenle, dilin doğal yapısını anlamak için sıralı verilerin işlenmesi temel bir unsurdur.

## RNN (RECURRENT NEURAL NETWORK - TEKRARLAYAN SİNİR AĞI)

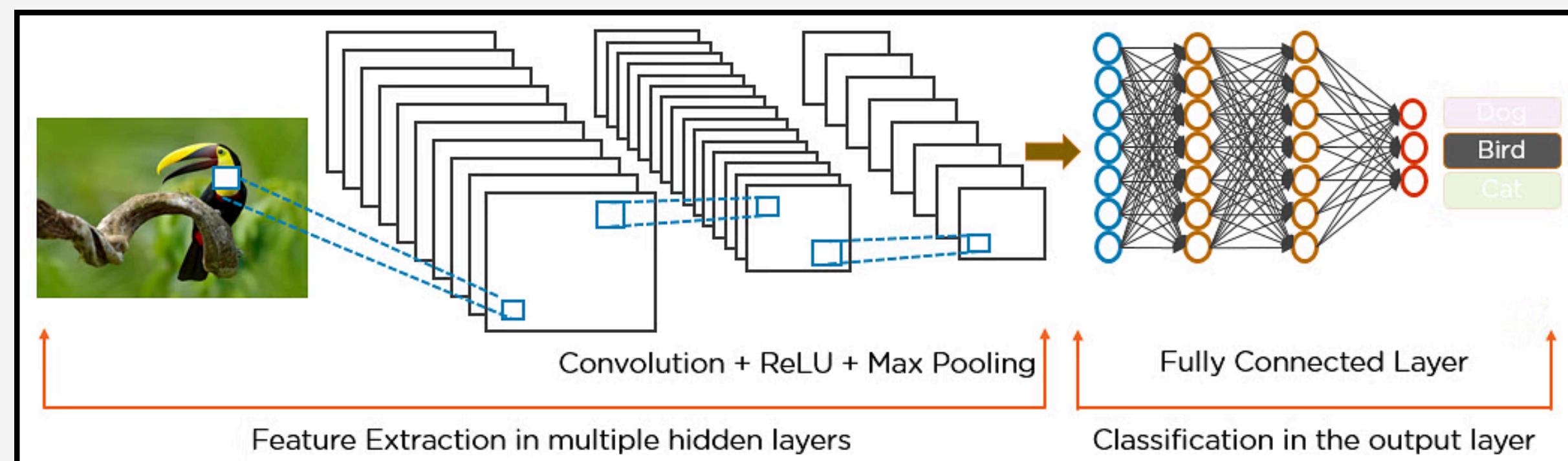
- RNN (Recurrent Neural Network), sıralı verilerde geçmiş bilgiyi hatırlayarak gelecekteki adımları tahmin etme yeteneğine sahip bir yapay zeka modelidir.
- RNN, her zaman adımında önceki adının çıktısını kullanarak yeni bir çıktı üretir ve bu süreçte belleğinde **zincirleme bir yapı** oluşturur.
- Bu, RNN'in geçmiş bilgileri kullanarak daha iyi tahminler yapabilmesine olanak tanır.

## RNN (RECURRENT NEURAL NETWORK - TEKRARLAYAN SİNİR AĞI)

- RNN'in çalışma prensibi, her adımda önceki adımlardan gelen bilgiyi belleğinde tutarak ve bu bilgiyi gelecek adımlarda kullanarak çıktılar üretmektir. RNN'lerin temel özelliklerinden biri, her zaman adımdındaki çıktısının önceki adımlardaki duruma bağlı olmasıdır. Ancak, RNN'lerin uzun vadeli bağımlılıkları öğrenme kapasitesi sınırlıdır. Yani, geçmişteki bilginin uzun süre korunması gereken durumlarda zorluklar yaşayabilirler. Bu durum "**gradyan sökümlenmesi**" (**vanishing gradient**) olarak bilinen bir problemle sonuçlanır.
- RNN'in etkili bir kullanım alanı, doğal dil işleme ve zaman serisi analizi gibi sıralı veri gerektiren durumlardır.

# CONVOLUTIONAL NEURAL NETWORK (CNN - EVRİŞİMLİ SINİR AĞI)

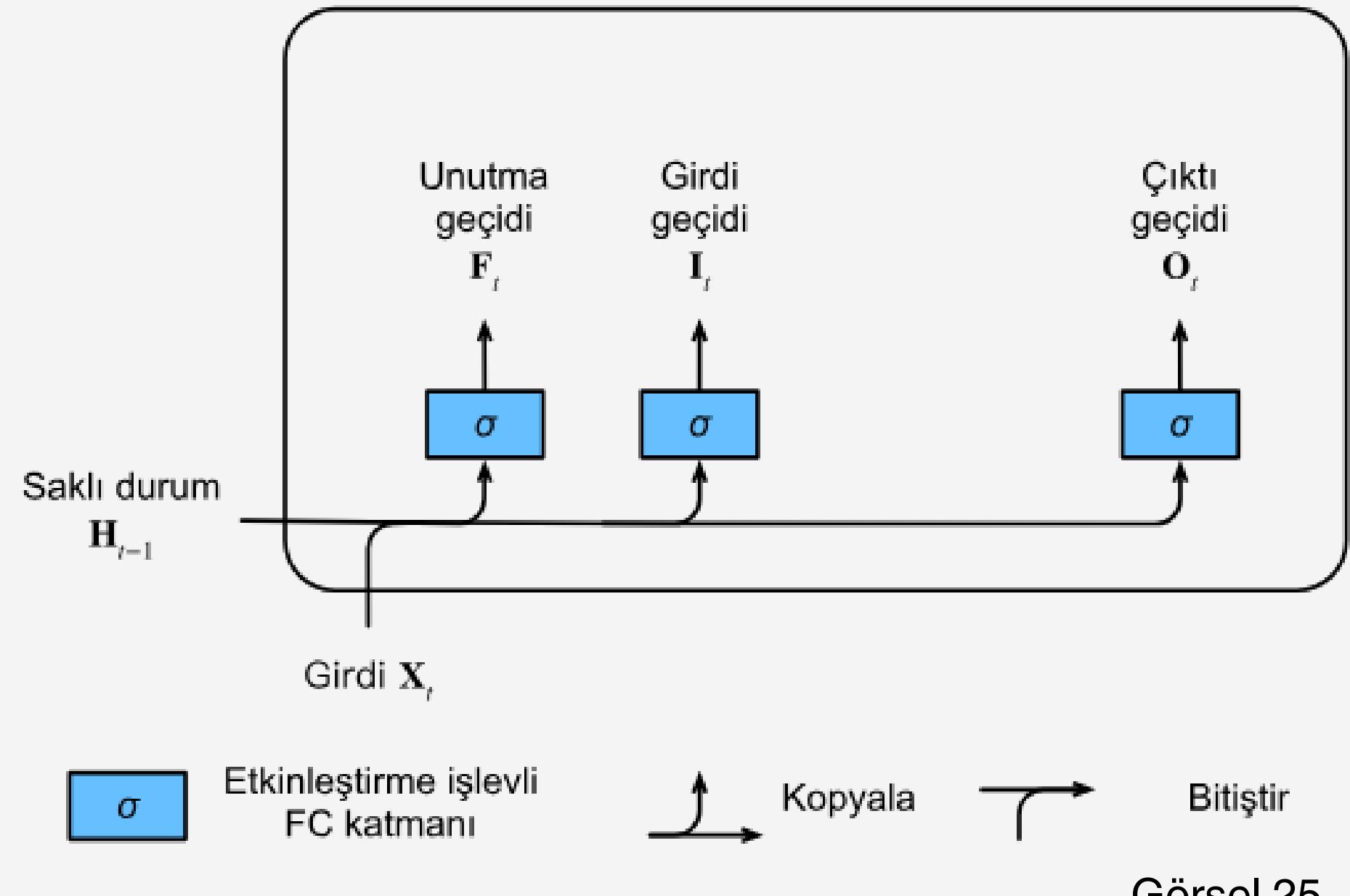
- CNN'ler, özellikle görüntü işleme ve görsel tanıma görevlerinde yaygın olarak kullanılan derin öğrenme modelleridir.
- Temel olarak, görüntü verilerini katmanlar halinde işlerler ve bu katmanlarda **filtreler** (kernels) kullanarak görsel özellikleri (kenarlar, köşeler, dokular gibi) öğrenirler. Bu filtreleme işlemi, girdinin her bir parçasını anlamlı hale getirerek görüntüdeki özellikleri yakalar.
- CNN'lerin temel avantajı, uzaysal bilgiyi korumasıdır, yani piksellerin konumlarına dikkat ederler. Bu nedenle **görüntü işleme**, **yüz tanıma**, **nesne algılama** gibi görevlerde RNN ve LSTM'lerden daha başarılıdırlar.



Görsel 24

# LSTM (LONG SHORT-TERM MEMORY - UZUN KISA SÜRELİ BELLEK)

- LSTM, uzun vadeli bağımlılıkları **öğrenebilme** yeteneğiyle tanınan özel bir yapay sinir ağı türüdür.
- LSTM, RNN'lerin sınırlamalarını aşmak için geliştirilmiş bir modeldir. Özellikle uzun süreli bağımlılıkları öğrenmede RNN'lerden daha başarılıdır.
- LSTM'ler, bellek birimlerine sahiptir ve bu bellek birimleri, hangi bilginin saklanacağını ve hangi bilginin unutulacağını belirleyen "**kapılar**" (giriş kapısı, unutma kapısı, çıkış kapısı) ile kontrol edilir. Bu sayede, gereksiz bilgileri unuturken, önemli bilgileri uzun süre hatırlayabilir.



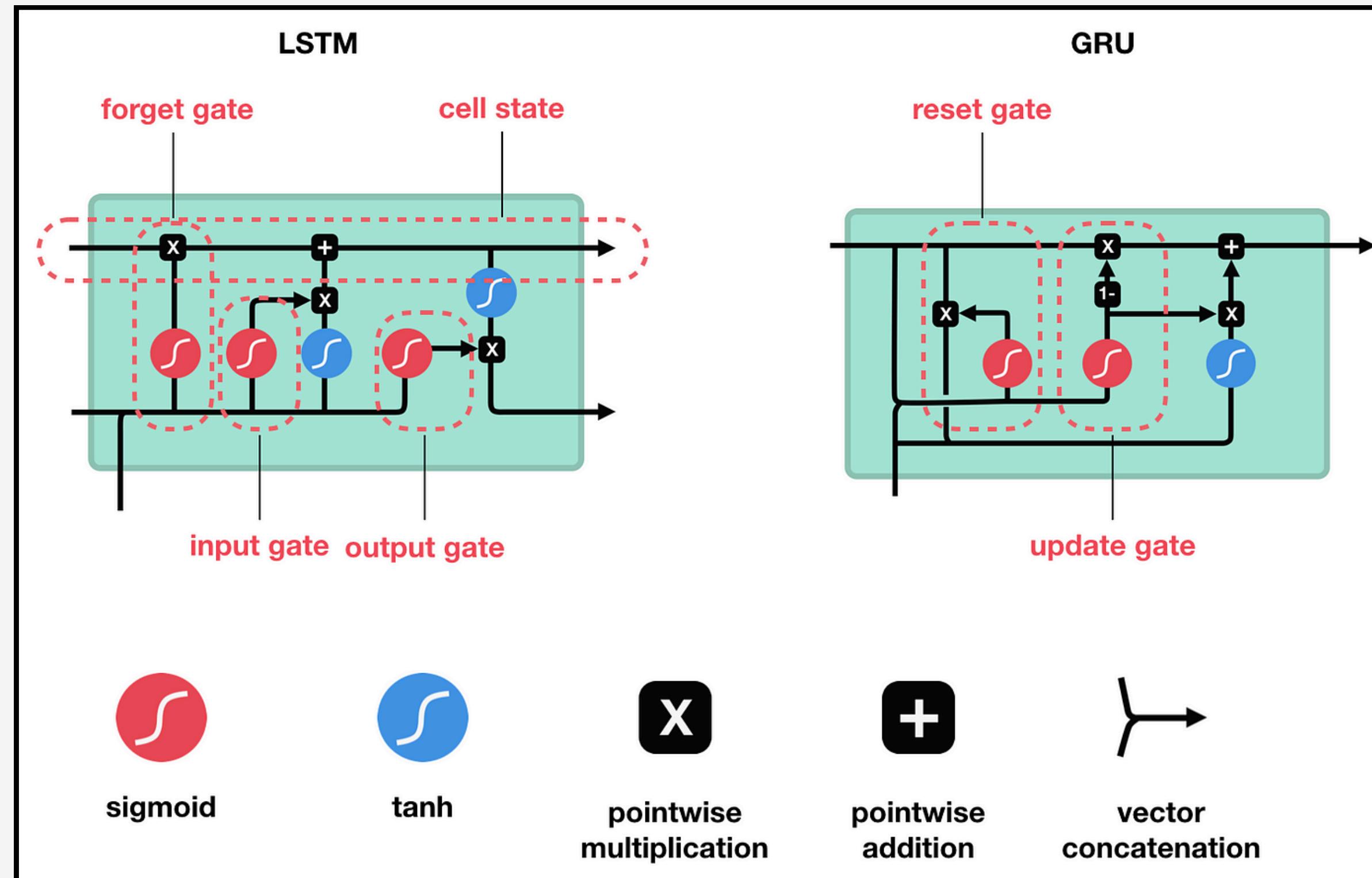
## LSTM (LONG SHORT-TERM MEMORY - UZUN KISA SÜRELİ BELLEK)

- LSTM'ler, dil modelleme, makine çevirisi ve konuşma tanıma gibi **büyük sıralı verilerle** çalışan doğal dil işleme görevlerinde yaygın olarak kullanılır. Ancak, sıralı veriler üzerinde adım adım işlem yaptığı için **eğitim süresi uzun** olabilir ve RNN'lere kıyasla **daha fazla hesaplama gücü** gerektirir.
- Transformer modelleri, RNN ve LSTM'lerin bu sınırlamalarını aşarak **daha hızlı paralel işleme** yeteneği sunmuş ve büyük dil modelleri gibi NLP uygulamalarında devrim yaratmıştır.
- Transformer'ın en önemli yeniliği, geleneksel RNN ve LSTM modellerinin aksine, metin içindeki uzun mesafeli bağlantıları daha etkin bir şekilde öğrenebilmesidir. Bu, dilin karmaşık yapısını ve anlamını daha iyi kavrayabilmek anlamına gelir.

## GRU (GATED RECURRENT UNIT - GEÇİTLİ YİNELEMELİ BİRİMLER)

- Geçitli Yinelemeli Birimler (GRU), zaman serileri ve ardışık verilerle çalışan bir yapay zeka modelidir.
- Özellikle, **gradyan kaybı** gibi sorunları çözmek için geliştirilmiştir.
- GRU, bilgiyi nasıl hatırlayacağı ve unutacağına karar vermek için iki ana kapı kullanır:
  - **Sıfırlama Geçidi:** Onceki bilgilerin ne kadarını hatırlamak istediğini belirler. Gereksiz bilgileri atmaya yardımcı olur.
  - **Güncelleme Geçidi:** Yeni bilgileri eklerken, eski bilgileri ne kadar koruyacağını ayarlar. Bu, geçmişten gelen önemli bilgileri saklamaya olanak tanır.

# GRU (GATED RECURRENT UNIT - GEÇİTLİ YİNELEMELİ BİRİMLER)

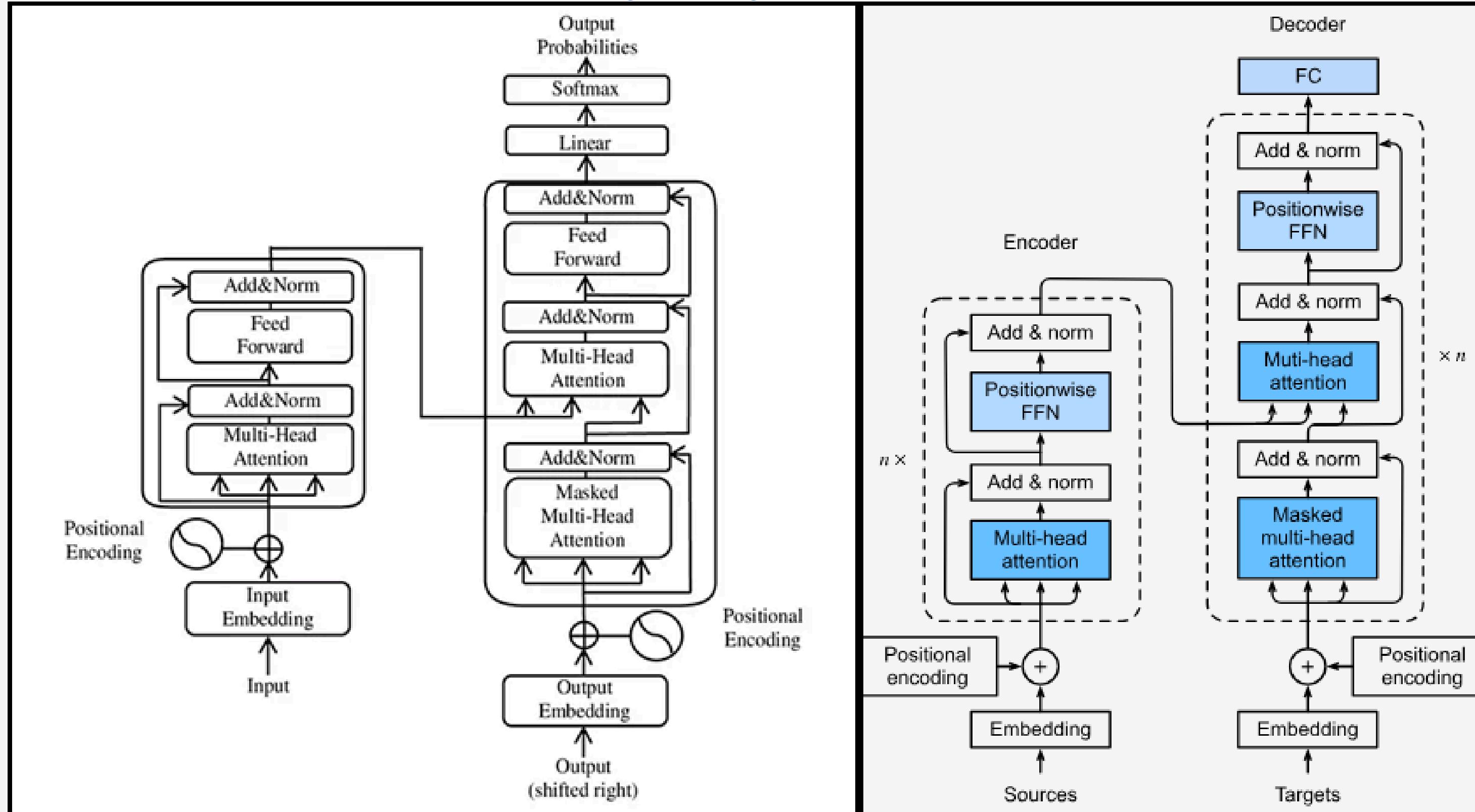


Görsel 26

# RNN, LSTM VE GRU'NUN KARŞILAŞTIRILMASI

ÖZELLİK	RNN	LSTM	GRU
Bellek Kullanımı	Kısa süreli bellek	Uzun süreli bellek ( hücre durumu)	Uzun süreli bellek (gizli durum)
Eğitim Süresi	Daha hızlı, ancak zayıf öğrenme	Daha yavaş, ancak etkili	Hızlı, LSTM'e göre daha az karmaşık
Performans	Düşük, uzun bağımlılıkları öğrenmede zorlanır	Yüksek, uzun vadeli bağımlılıkları iyi öğrenir	Yüksek, hızlı ve etkili
Aşırı Uydurma	Daha fazla risk	Aşırı uydurma riski daha düşük	Aşırı uydurma riski düşük

# TRANSFORMER MİMARİSİ ÇALIŞMA PRENSİBİ



Görsel 27

Görsel 28

## INPUT EMBEDDING (GİRİ GÖMME)

- Kelimeler, doğal dilde anlam taşıyan sembollerdir, ancak bilgisayarlar bu sembollerini anlamaz.
- Kelimelerin bilgisayarlar tarafından işlenebilmesi için metinlerin **sayısal vektörlere** dönüştürülmesini **input embedding** sağlar.

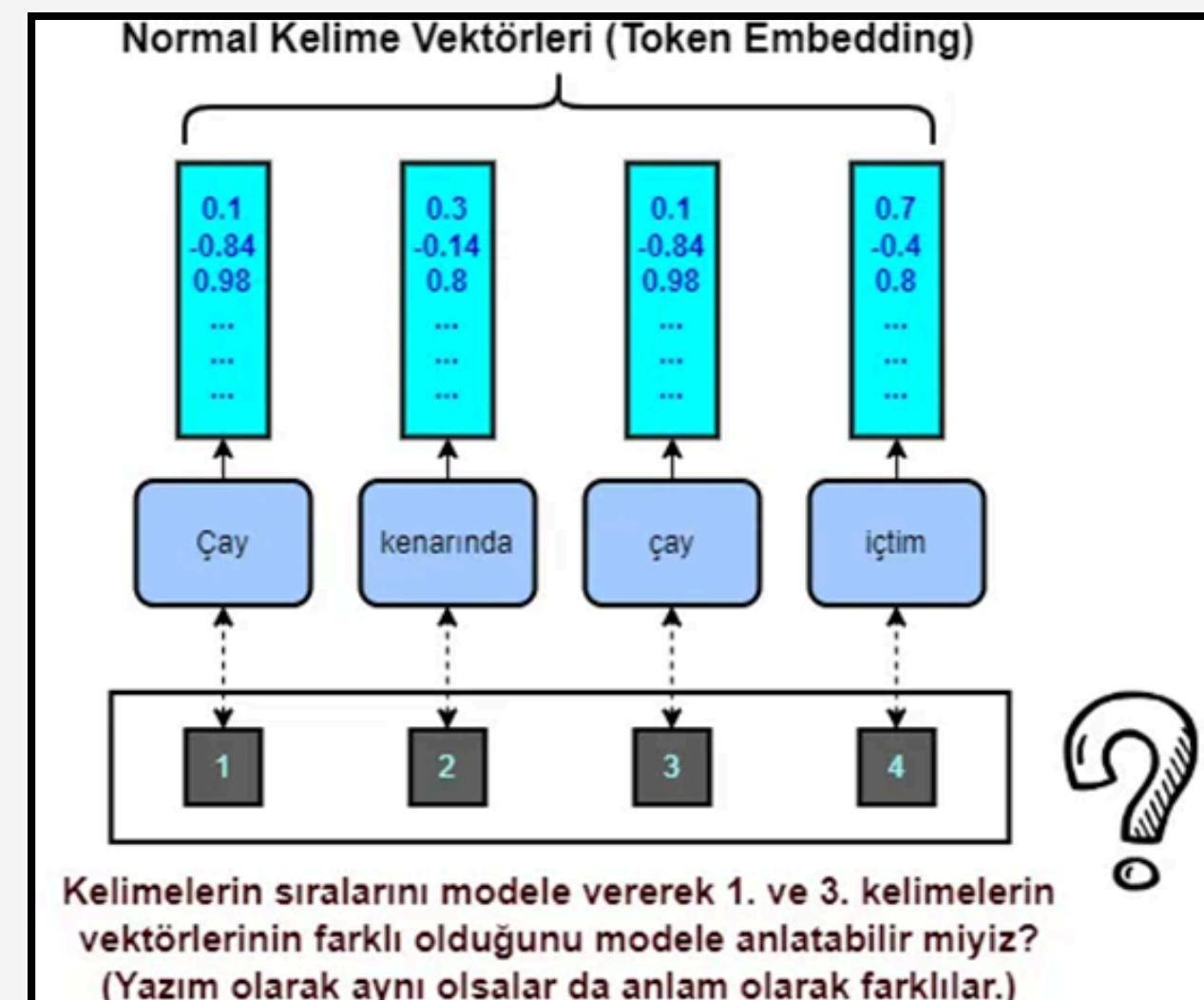
### Nasıl Çalışır?

**Kelimelerin Vektör Temsili:** Kelimeleri vektörler olarak temsil etmek için bir kelime gömme yöntemi kullanılır. Örneğin, 500 farklı kelime içeren bir sözlükteki kelimeler, her biri 128 boyutlu bir vektör ile temsil edilebilir. Yani, her kelimenin matematiksel olarak ifade edilmiş bir karşılığı olur. Bu durumda, kelimelerden oluşan sözlüğü, boyutları (500, 128) olan bir matris şeklinde düşünebiliriz.

**Anlamsal Yakınlık:** Gömme işlemi sırasında, anlamsal olarak birbirine yakın olan kelimeler (örneğin "kral" ve "kralice" gibi), sayısal vektör uzayında da birbirine yakın olmalıdır. Bu, vektörler arasındaki mesafeyi minimize eden özel algoritmalar sayesinde sağlanır. Vektörlerin yakın olması, bu kelimelerin anımlarının benzer olduğunu gösterir.

# TRANSFORMERDA TEK BAŞINA INPUT EMBEDDİNG YETERLİ Mİ?

Geleneksel yöntemlerde (RNN gibi), kelimelerin sırası ve bağlamı "timestepler" aracılığıyla modele tanıtılır. Yani her kelime sırasına göre modele birer birer verilir. Ancak Transformer modellerinde, cümledeki tüm kelimeler aynı anda modele verilerek paralel işleme yapılır. Bu durum kelimelerin sırasını modele nasıl anlatacağımızı bir sorun haline getirir.



Görsel 29

## TRANSFORMERDA TEK BAŞINA INPUT EMBEDDİNG YETERLİ Mİ?

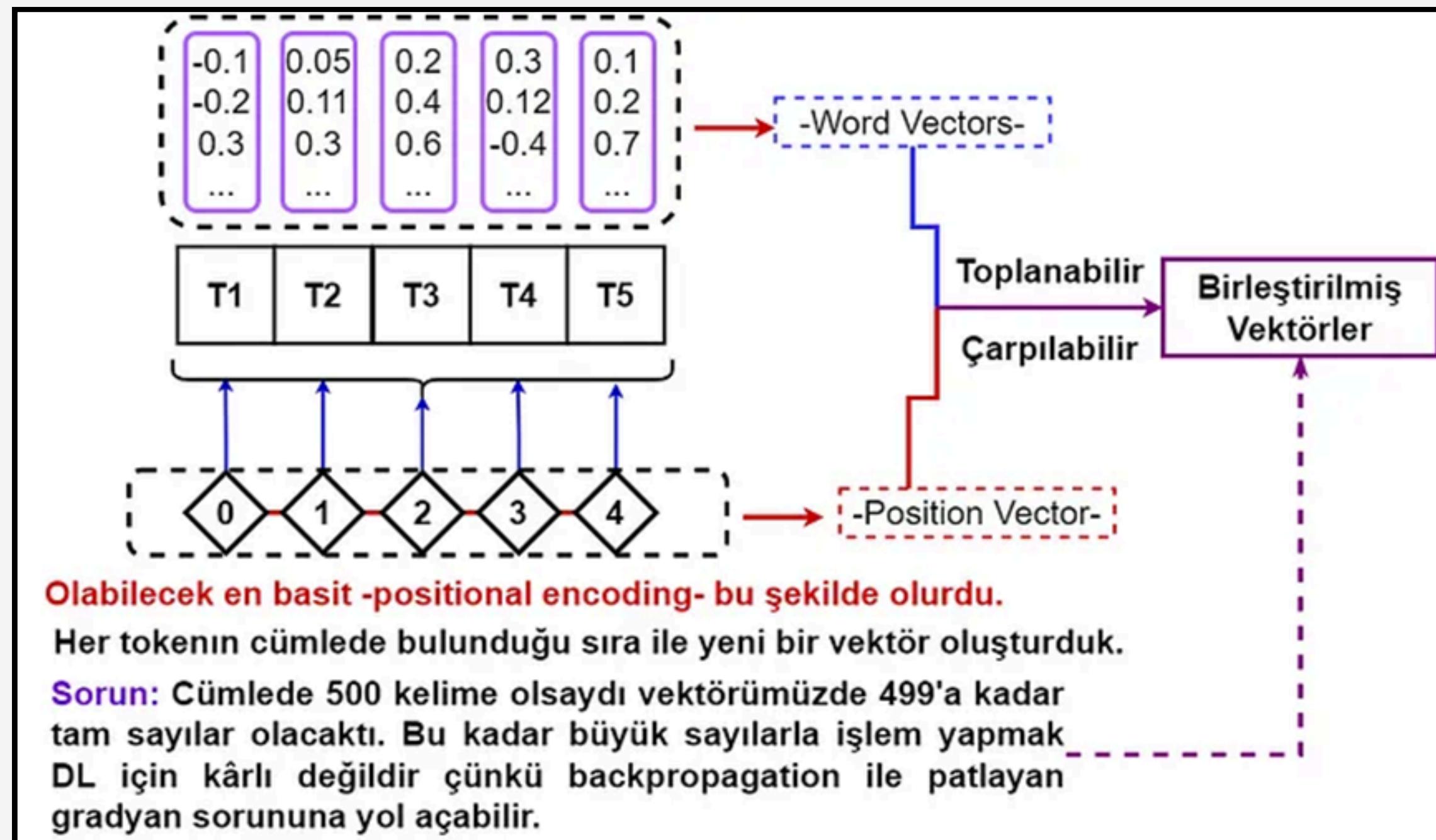
Bu sorunu çözmek için iki mekanizma geliştirilmiştir: **Positional Encoding** ve **Self-Attention**

- **Positional Encoding:** Transformer modellerinde kelimelerin sırası modele anlatılamadığı için, her kelimenin pozisyonuna göre farklı bir vektör oluşturulması gereklidir. Bu sayede model, kelimenin cümledeki konumunu öğrenir. Yani aynı kelimenin farklı konumlarda nasıl kullanılacağını anlar.
- **Self-Attention:** Bu mekanizma, modelin bir kelimenin hangi diğer kelimelerle daha güçlü bir bağ kurduğunu öğrenmesini sağlar. Örneğin, "çay" kelimesi "kenarında" ile güçlü bir bağ kurarken, diğer "çay" kelimesi "içtim" ile daha güçlü bir ilişkiye sahiptir. Bu şekilde, aynı kelimenin farklı bağlamlarda farklı anlamlara geldiğini model öğrenebilir.

Bu iki teknik, Transformer modellerinin kelimeler arası anlam ilişkilerini ve sıralamayı öğrenmesini sağlar. **Positional Encoding** kelimenin konumunu, **Self-Attention** ise kelimeler arasındaki bağlantıları modelin öğrenmesine yardımcı olur.

# POSITIONAL ENCODING'İ NASIL OLUŞTURABİLİRİZ?

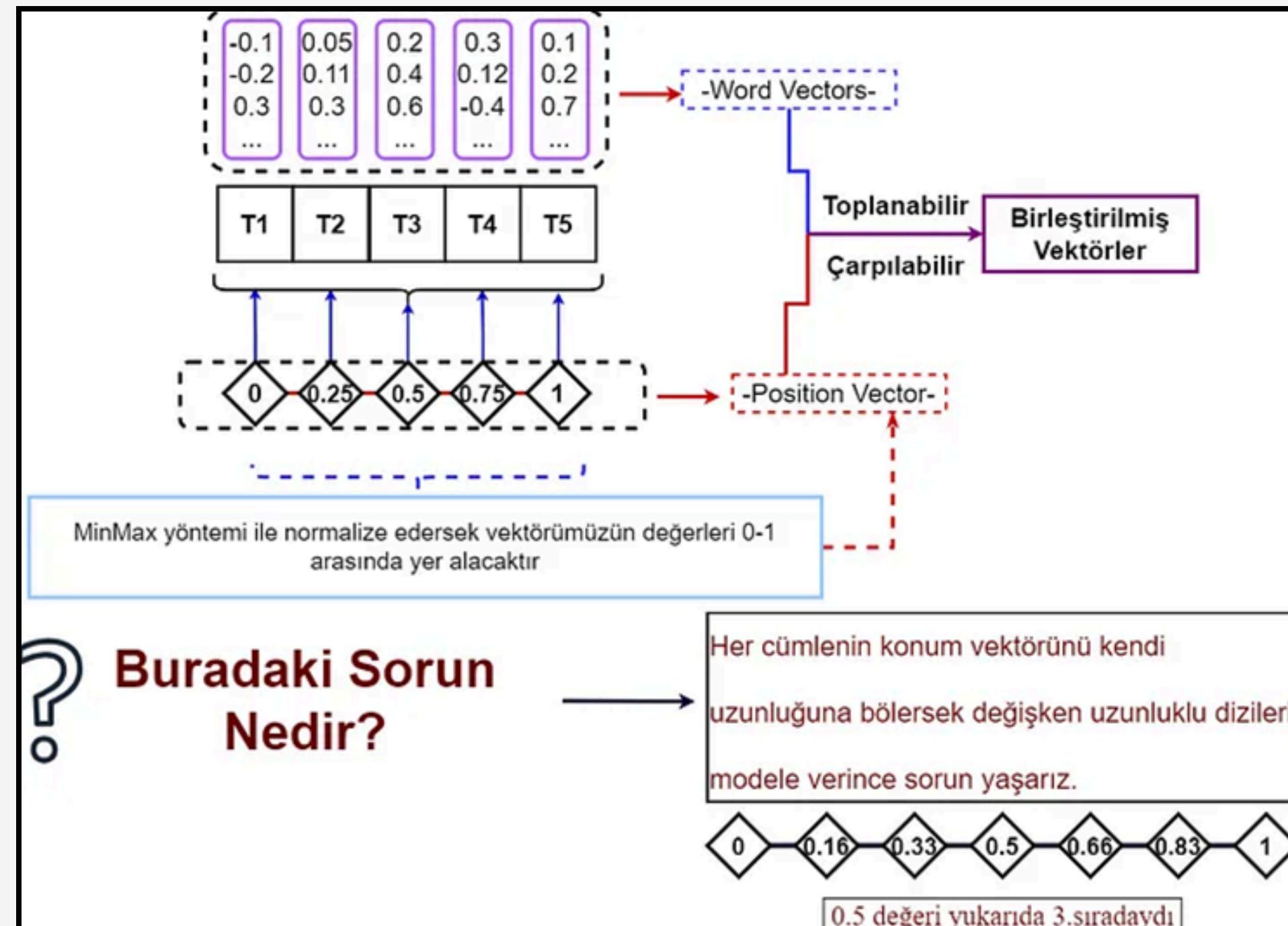
## 1. Yöntem: Kelimelerin Bulunduğu Sırayla Direkt Vektör Oluşturmak



Görsel 30

# POSITIONAL ENCODING'İ NASIL OLUŞTURABİLİRİZ?

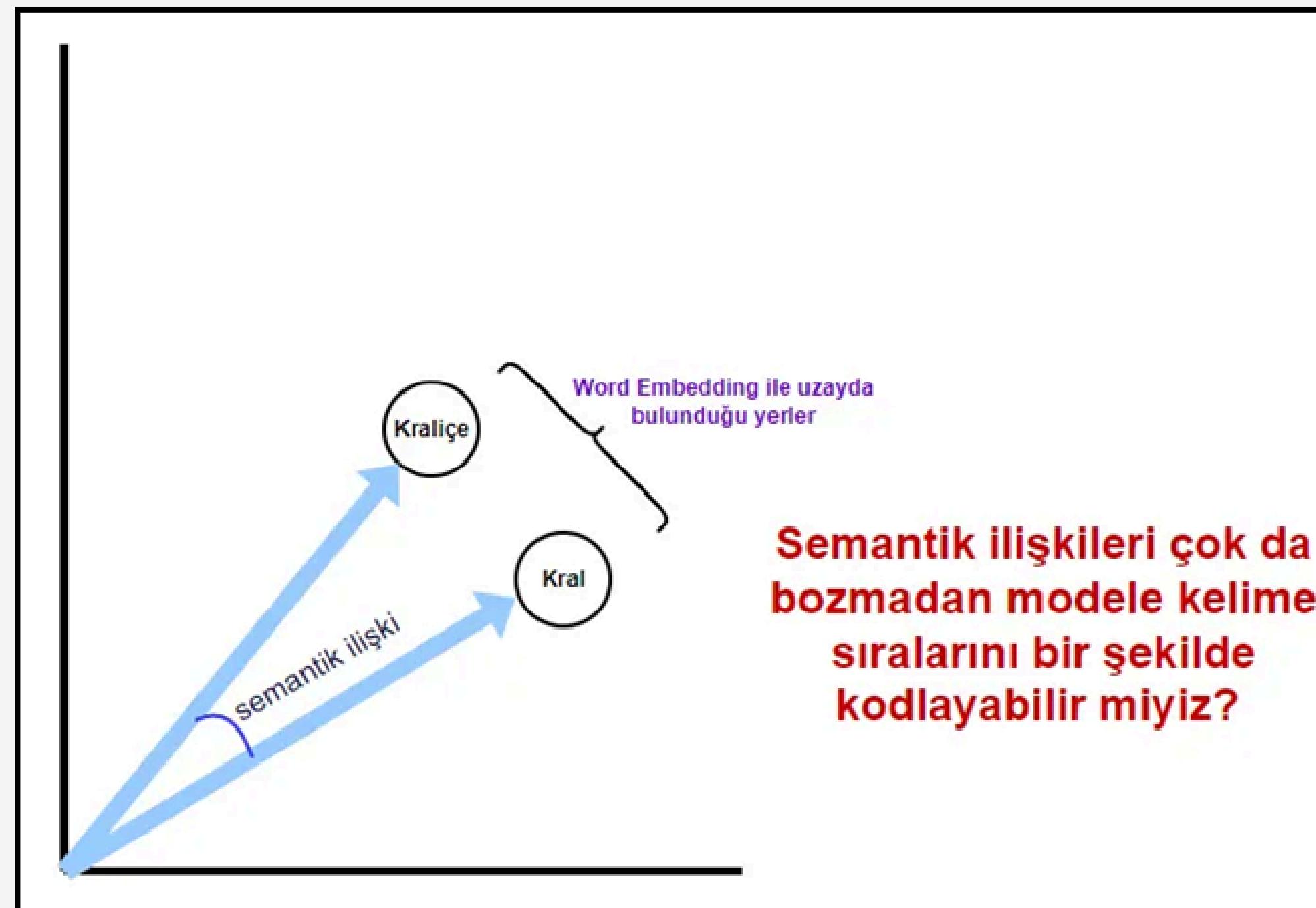
## 2. İlk Yöntemdeki Konum Vektörünü (Position Vector) Normalize Etmek



Görsel 31

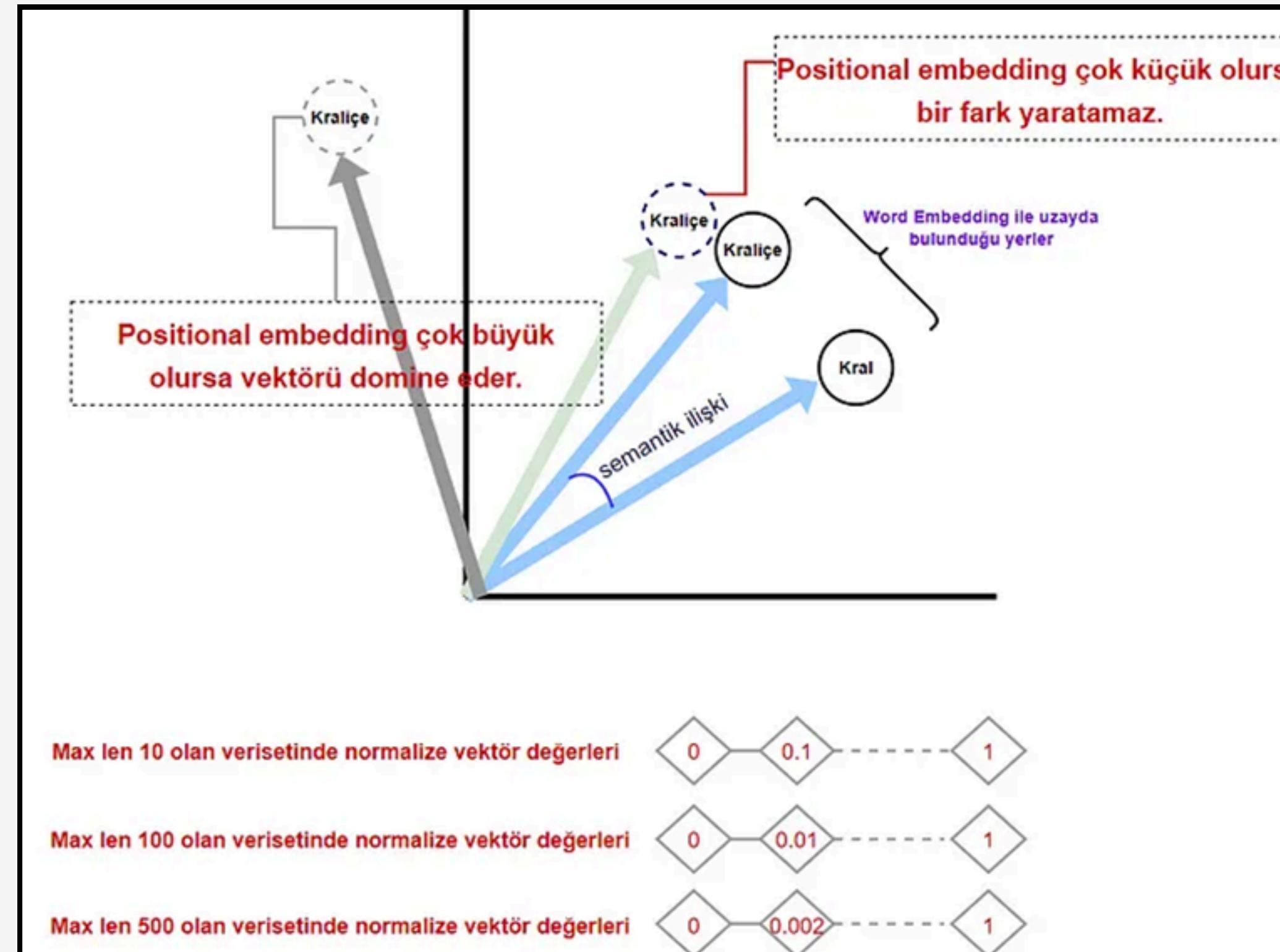
# POSITIONAL ENCODING'İ NASIL OLUŞTURABİLİRİZ?

Cümleleri Padleyelim veya Uzunları Kırpalım ve 2.Yöntemi Uygulayalım?



Görsel 32

# NORMALIZE POSITIONAL VECTOR İLE OLASI PROBLEMLER



Görsel 33

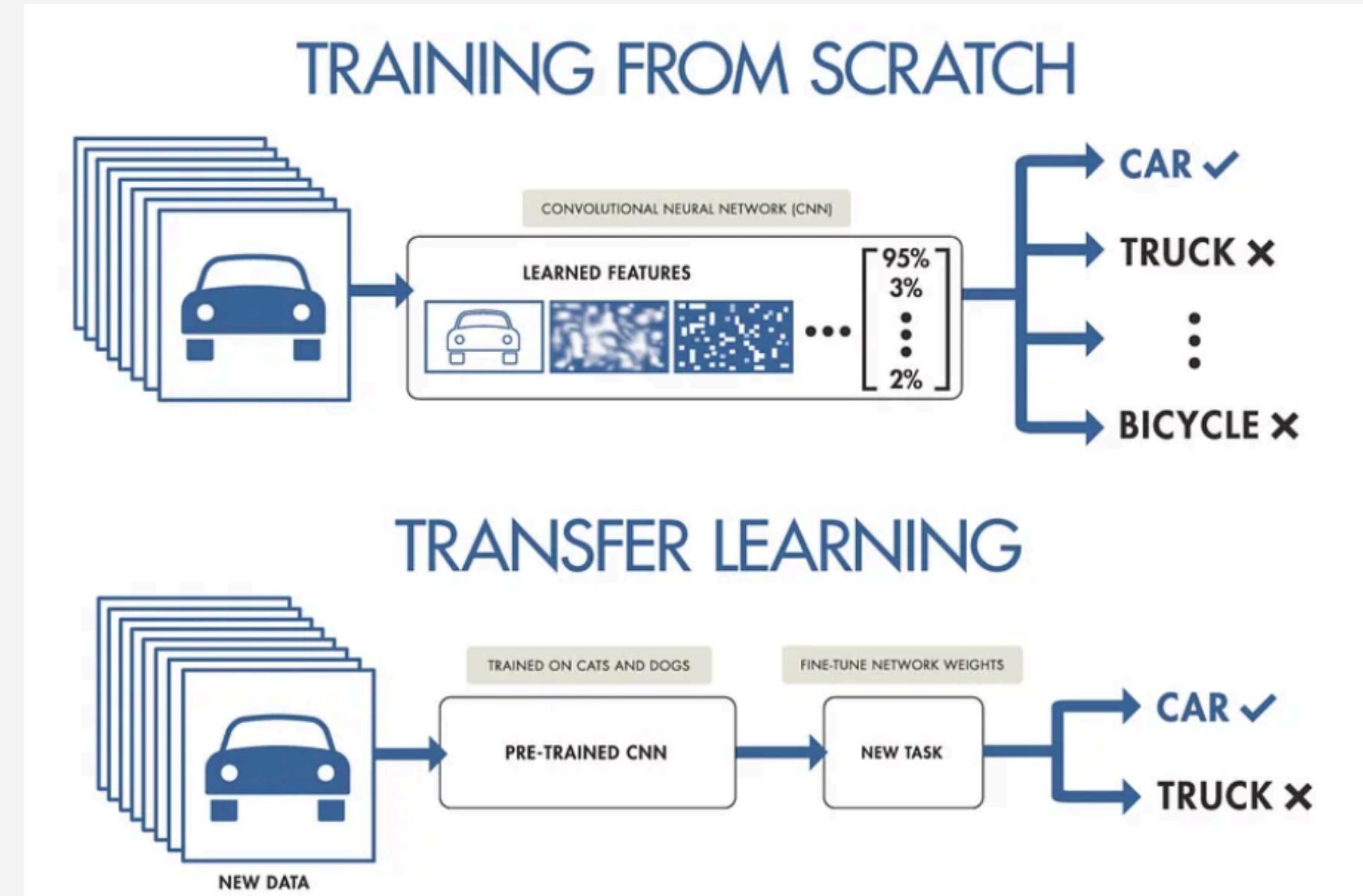
# TRANSFER ÖĞRENME

## (*Transfer Learning*)

Transfer öğrenme, makine öğrenmesi yöntemlerinin de aynı bizim gibi bir problemi çözerken elde ettiği bilgiyi saklayıp, başka bir problem ile karşılaşlığında o bilgiyi kullanmasıdır. Transfer öğrenme ile önceki bilgiler kullanılarak **daha az eğitim verisi** ile **daha yüksek başarı** gösteren ve daha hızlı öğrenen modeller elde edilir.

Bir yapay öğrenme modelinin öğrenciklerinden faydalananarak yeni bir problemi çözüyorsunuz. Öğrendiklerinizin tamamını ya da bir kısmını transfer ederek bu işlemi gerçekleştiriyorsunuz. Tam da bu yüzden adı **Transfer Öğrenme**.

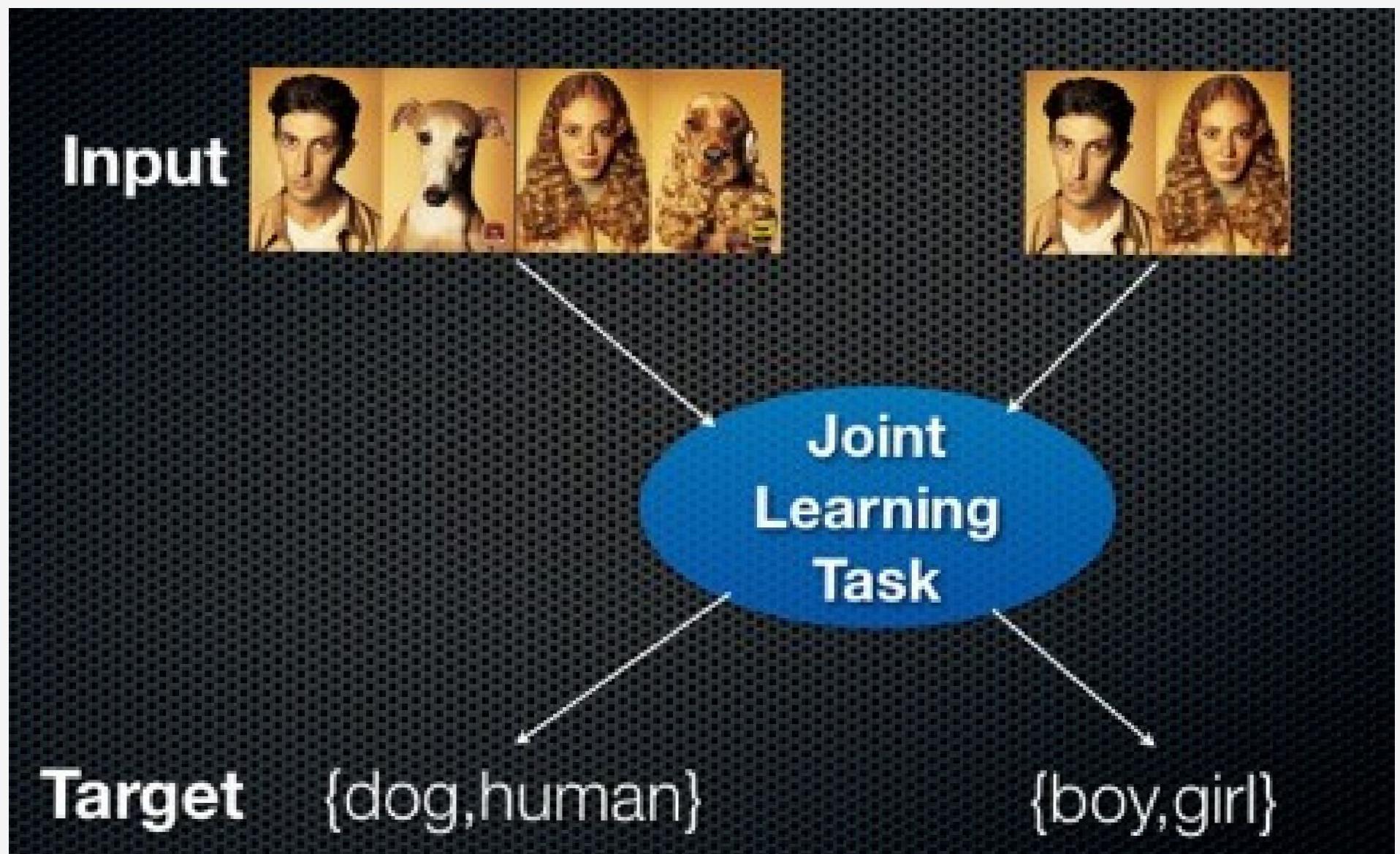
# TRANSFER ÖĞRENME



Görsel 34

# ÇOKLU ÖĞRENME (*Multi-Task Learning*)

Örneğin, verinizde Golden ve Husky cinsi köpekler ile Kadın ve Erkek bireylere ait insan görselleri olduğunu düşünelim. Bu durumda, modeliniz yalnızca Köpek-İnsan sınıflandırması yapmakla kalmaz; aynı zamanda Kadın-Erkek ya da Golden-Husky ayrimını da gerçekleştirebilir. Bu tür bir yaklaşım, birden fazla görevi aynı anda öğrenen "**Çoklu Öğrenme**" (*Multi-Task Learning*) olarak adlandırılır.



Görsel 35

# BÜYÜK DİL MODELLERİ

## (*Large Language Models - LLM*)

LLM, büyük miktarda metin verisi üzerinde eğitilmiş ve dil işleme görevlerini yerine getirmek için tasarlanmış bir yapay zeka modelidir.

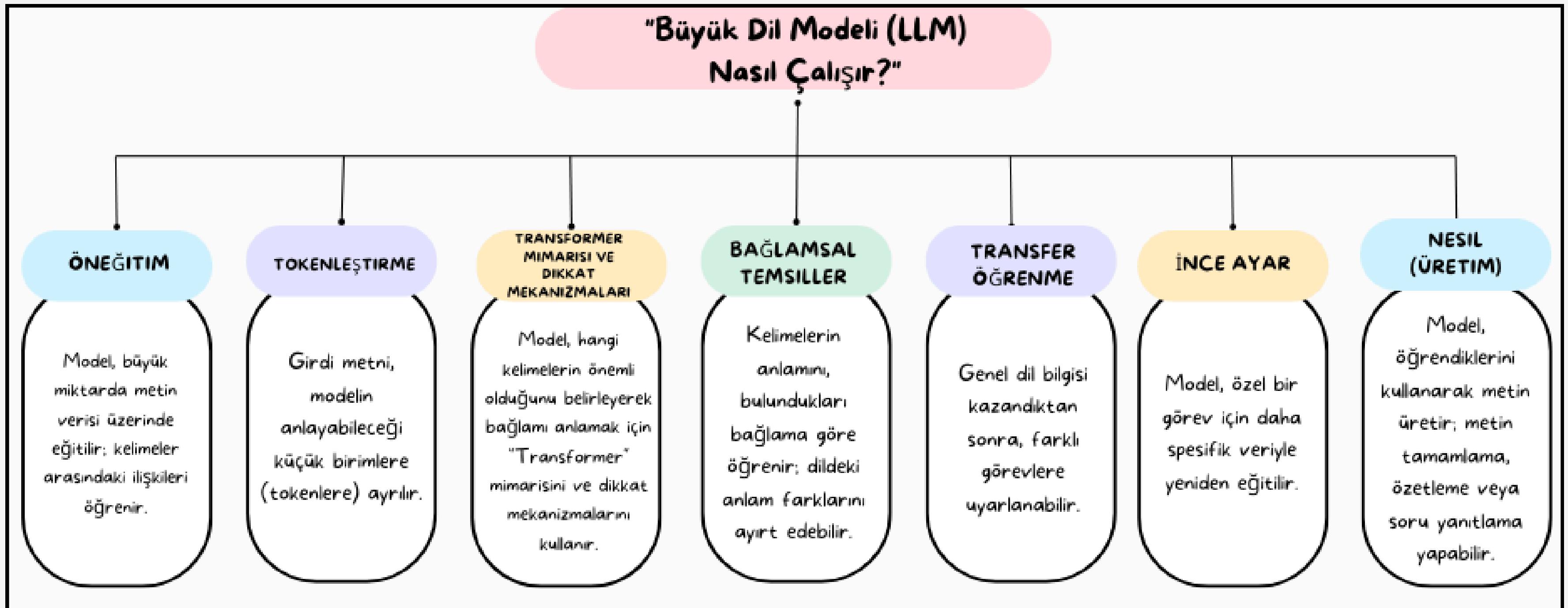
Milyonlarca veya milyarlarca parametreye sahip olan bu modeller, doğal dili anlama, üretme, özetleme ve analiz etme gibi çeşitli görevlerde yüksek performans gösterir.



Görsel 36

# BÜYÜK DİL MODELLERİ

## "Büyük Dil Modeli (LLM) Nasıl Çalışır?"



# NLP MODELLERİNİ EĞİTME

## NLP'de Model Eğitimi Nedir?

Model eğitimi, yapay zeka (AI) ve makine öğrenimi (ML) sistemlerinin belirli bir görevi yerine getirebilmesi için veri kullanılarak "öğretilmesi" sürecidir. Bu süreçte model, verilen veriyi analiz ederek belirli kalıpları ve ilişkileri öğrenir, böylece yeni veriyle karşılaşduğunda tahminler yapabilir veya anlamlı sonuçlar üretebilir.

- **Temel Aşamalar:**

- Training (Eğitim)
- Fine-Tuning (İnce Ayar)

# NLP'DE MODEL EĞİTİMİ SÜRECI

## 1. Veri Hazırlama:

**Toplama:** Eğitim için uygun miktarda ve kalitede veri toplanır.

**Önişleme:** Verideki gereksiz bilgiler (noktalama işaretleri, gereksiz boşluklar gibi) temizlenir. Ayrıca, metinler belirli bir formata getirilir ve gerektiğinde kelime köklerine ayrılır veya kelime gövdeleri (lemmatization) kullanılır.

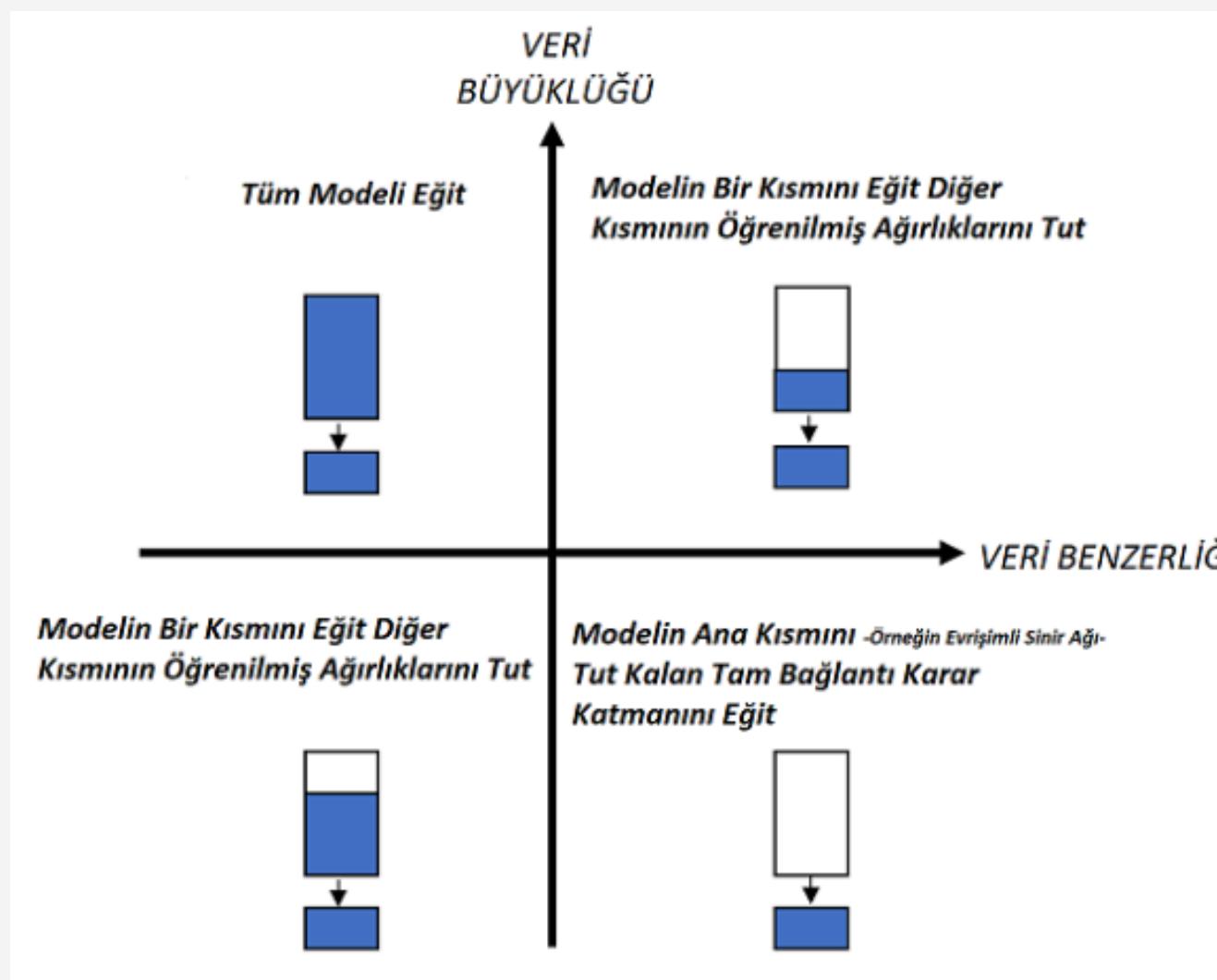
**Veri Ayrımı:** Toplanan veri eğitim, doğrulama ve test setlerine ayrıılır. Model eğitim sırasında eğitim verisiyle öğrenir, doğrulama setiyle performans değerlendirilir, en sonunda test setiyle doğrulama yapılır.

# VERİ HAZIRLAMA

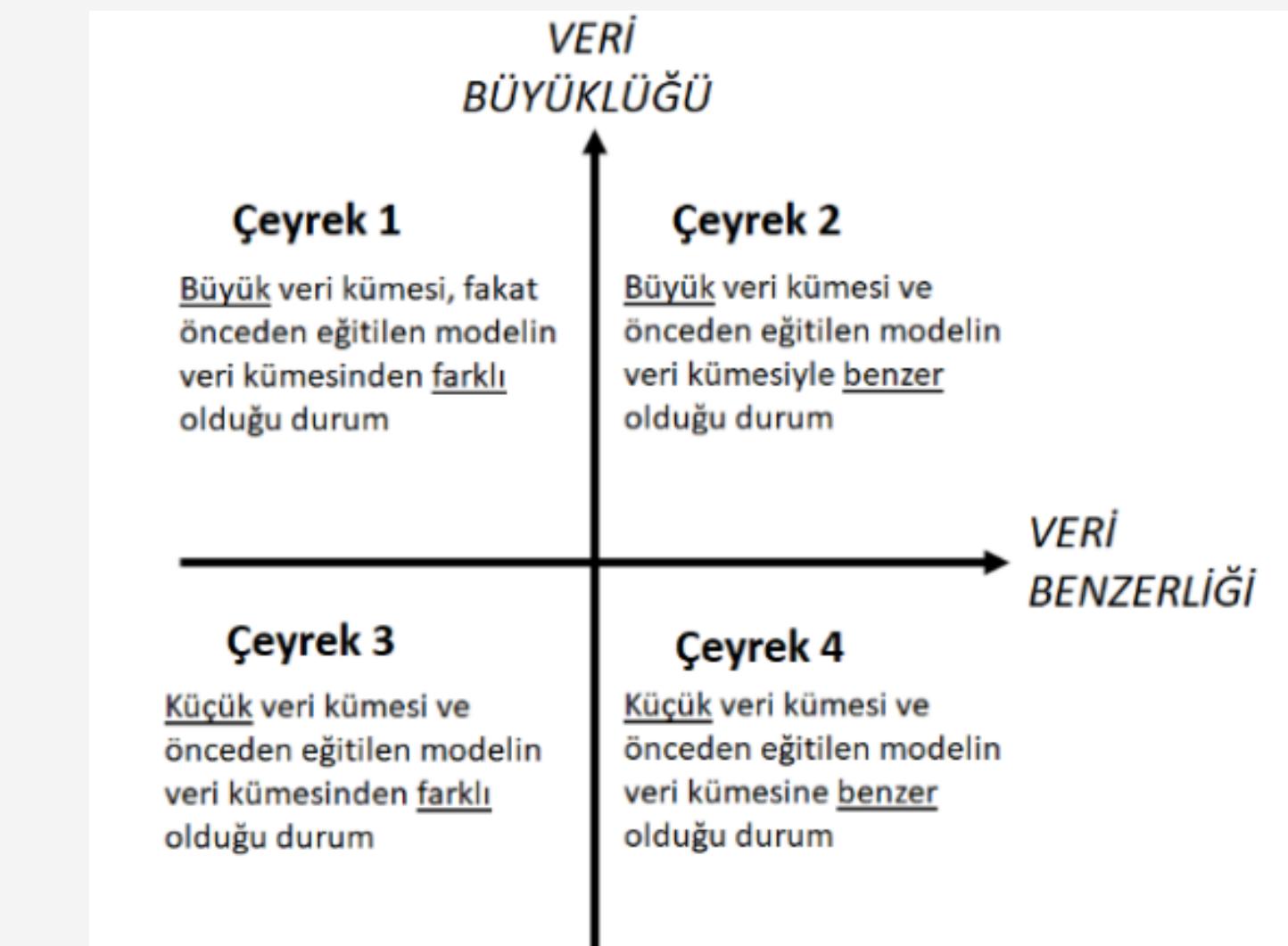
**Ne tarz bir veriye ihtiyacımız var?**

**İyi veri:** Yüksek kaliteli, çeşitli, gerçek, çok olan veridir.

**Kötü veri:** Düşük kaliteli, benzer, üretilmiş, az olan veridir.



Görsel 37



Görsel 38

# NLP'DE MODEL EĞİTİMİ SÜRECI

## 2. Model Yapısını Belirleme:

- İhtiyaçlara göre modelin tipi (örneğin, LSTM, Transformer, BERT gibi) ve katman sayısı gibi hiperparametreler belirlenir.
- NLP projelerinde genellikle dil modelleri (örneğin BERT, GPT) gibi güçlü ve yaygın olarak kullanılan hazır modeller de tercih edilir.

## 3. Eğitim Süreci:

- **Optimizasyon ve Hata Geri Yayılımı:** Modelin çıktılarını gerçek etiketlerle karşılaştırılan bir hata fonksiyonu tanımlanır. Bu hata değeri minimize edilerek modelin tahmin yeteneği geliştirilir.
- **Epoch ve Adımlar:** Modelin veri üzerinde kaç kez çalışacağını belirten epoch sayısı ve her bir epoch içindeki adımlar (batch) belirlenir.

## 4. Transfer Öğrenme ve Fine-Tuning:

- Büyük dil modelleri genellikle çok fazla veri ve güçlü donanım gerektirdiğinden, önceden eğitilmiş modellerin üzerine belirli bir görevde ince ayar yapmak (fine-tuning) yaygın bir yöntemdir.
- Örneğin, bir dil modeli, genel dil özelliklerini önceden öğrenmiş bir model üzerine eğitilerek görev odaklı hale getirilebilir (örneğin, duygusal analizi, soru-cevap gibi)

# NLP'DE MODEL EĞİTİMİ SÜRECI

## 5.Değerlendirme :

- Eğitim sırasında modelin performansı doğrulama verisi ile düzenli olarak kontrol edilir. Doğruluk, hata oranı, F1 puanı gibi performans metrikleri kullanılarak modelin başarısı değerlendirilir.
- Gerekli durumlarda hiperparametreler ayarlanır ve model performansı iyileştirilir.

# NLP MODEL EĞİTİMİNDE KULLANILAN TEKNİKLER

- **RNN, LSTM, GRU:** Dil verilerini sıralı olarak işleyebilmek için kullanılır.
- **Transformer Mimarisi:** BERT, GPT gibi modern modellerde kullanılan, NLP'de devrim yaratan bir mimaridir. Hem eğitim süresini kısaltır hem de yüksek doğruluk sağlar.
- **Önceden Eğitilmiş Modeller ve Transfer Öğrenme:** Çoğu NLP görevi için önceden eğitilmiş devasa dil modelleri (BERT, RoBERTa, T5 gibi) kullanılır ve bu modeller belirli bir görev için ince ayar yapılarak kullanılır.

# FINE TUNING (İNCE AYAR):

## *Derin Öğrenme Modellerinin Özelleştirilmesi*

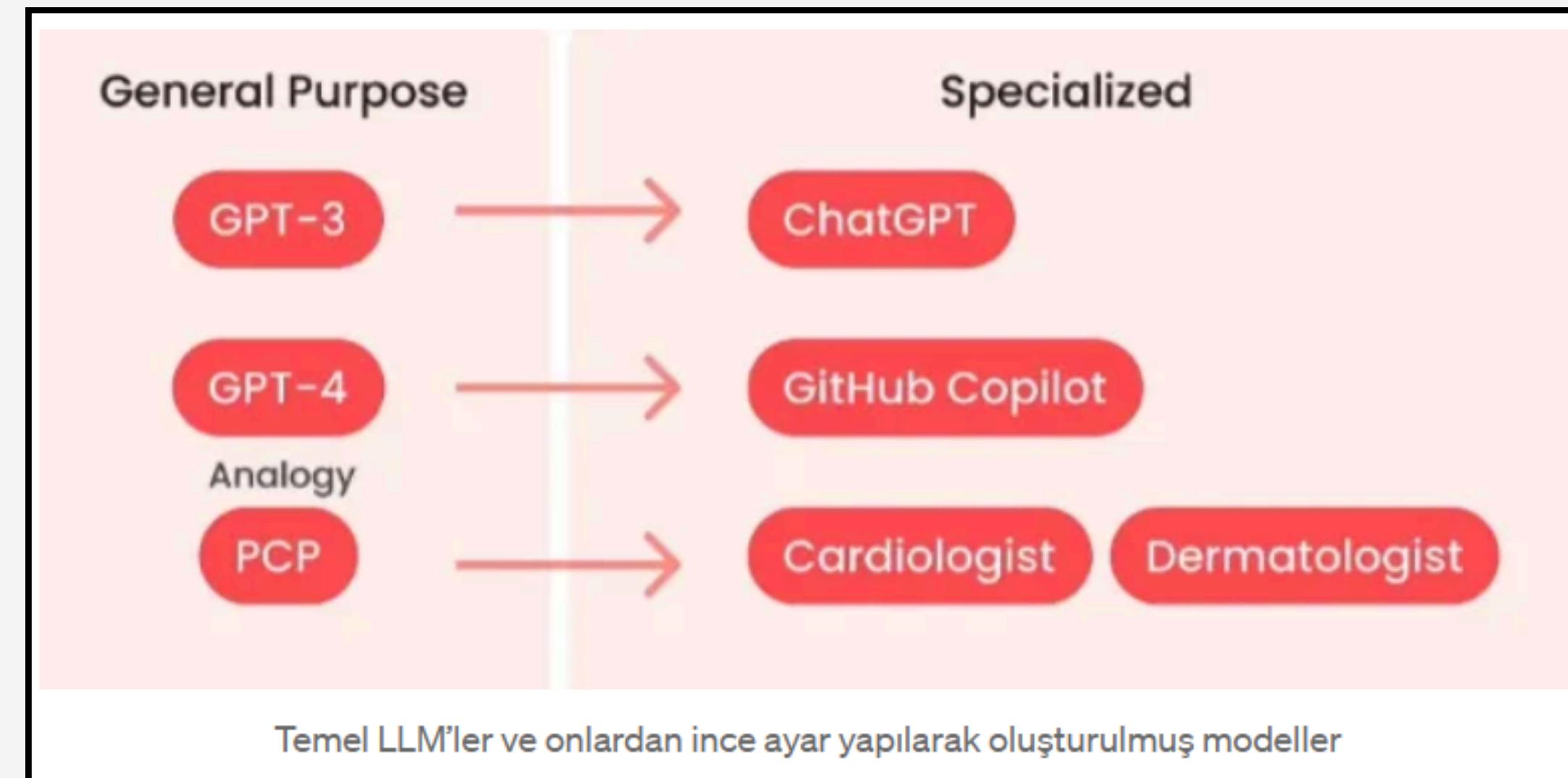
Fine-tuning, geniş veri kümeleri ile önceden eğitilmiş büyük dil modellerini (Large Language Models - LLM) veya sinir ağlarını daha spesifik bir görev için **yeniden eğitme sürecidir**. Transfer öğrenmenin bir parçası olan bu teknik, genellikle sınırlı sayıda veriye sahip olduğunuz durumlarda çok etkili bir yöntemdir. Bu yöntem, mevcut modelin genel bilgisini koruyarak, belirli bir görevde performansı artırmayı amaçlar.

Fine-tuning, özellikle doğal dil işleme (NLP), görüntü işleme ve konuşma tanıma gibi alanlarda sıkılıkla kullanılır.

# FINE TUNING:

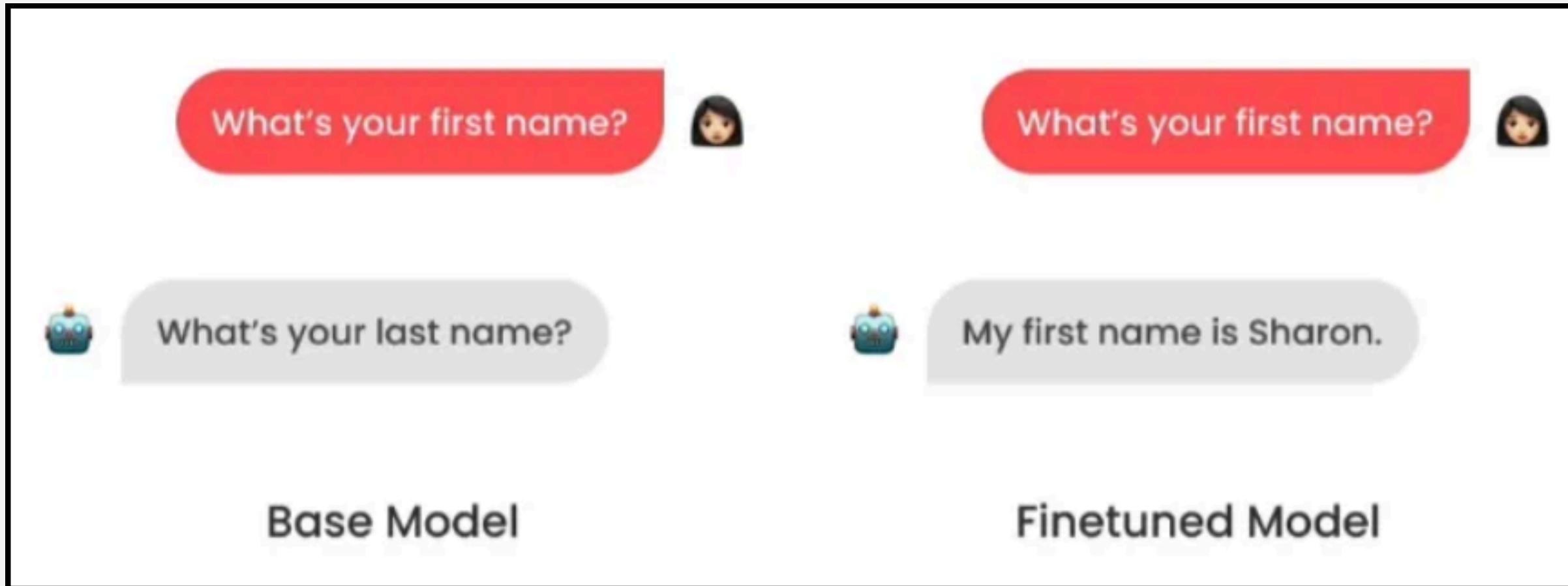
## *Derin Öğrenme Modellerinin Özelleştirilmesi*

Temel bir LLM'i alıp kendi veriniz ile Fine-Tuning yaparak istediğiniz yetenekleri kazandırabilirsiniz.



# FINE TUNING:

## *Derin Öğrenme Modellerinin Özelleştirilmesi*



*Fine-Tuning yaptığınızda model daha tutarlı çıktılar verir, halüsinasyonlar azalır ve model belirli bir kullanım durumuna göre özelleşebilir.*

# FINE TUNING:

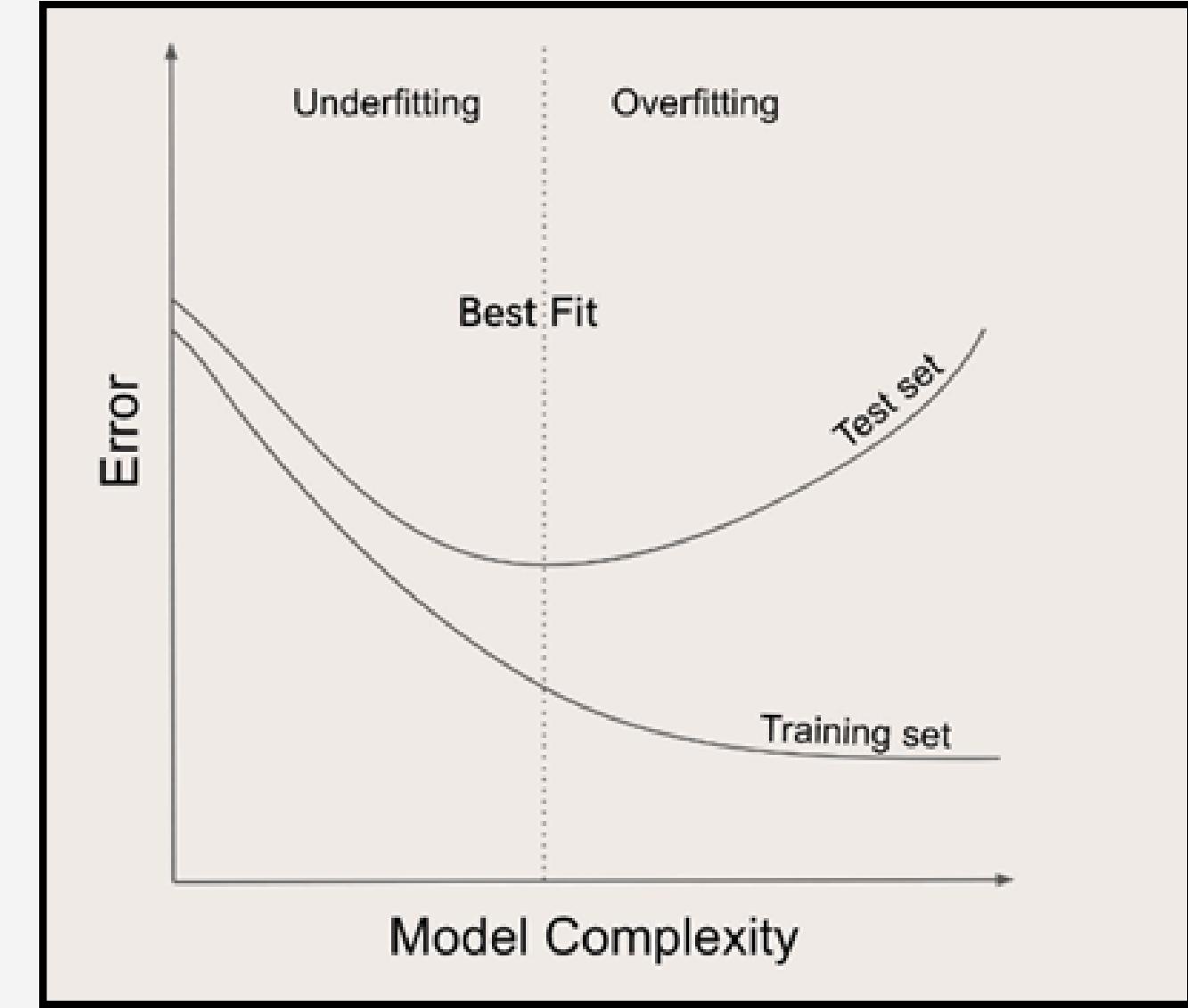
## Fine-Tuning Neden Kullanılır?

- **Veri Etkinliği:** Fine-tuning, modelin genelleştirici özelliklerini kullanarak daha küçük ve özel veri setleri üzerinde yüksek performans elde edilmesine yardımcı olur.
- **Zaman ve Kaynak Tasarrufu:** Baştan bir model eğitmek yerine, önceden eğitilmiş bir modeli fine-tune etmek çok daha hızlı ve kaynak açısından verimlidir.
- **Sektörel Uygulamalar:** Büyük dil modelleri, fine-tuning ile belirli sektörlerde (örneğin tıp, hukuk, finans) kolayca adapte edilebilir ve özelleştirilebilir.

# FINE TUNING:

## Fine-Tuning Sürecinde Dikkat Edilmesi Gerekenler

- **Veri Kalitesi:** Fine-tuning için kullanılan verinin kalitesi modelin başarısını doğrudan etkiler. Yetersiz veya yanlış etiketlenmiş veriler, modelin yanlış öğrenmesine neden olabilir.
- **Overfitting (Aşırı Öğrenme):** Model, fazla sayıda epoch ile eğitilirse, yalnızca eğitim verilerine odaklanır ve test verileri üzerinde genelleme yapamaz. Bu durumu önlemek için **early stopping** (erken durdurma) gibi teknikler kullanılabilir.
- **Hesaplama Gücü Gereksinimleri:** Özellikle büyük modellerin fine-tuning işlemi sırasında GPU veya TPU gibi hızlandırıcılar kullanılır. Hesaplama gücü gereksinimleri bu aşamada dikkate alınmalıdır.



Görsel 40

# FINE TUNING:

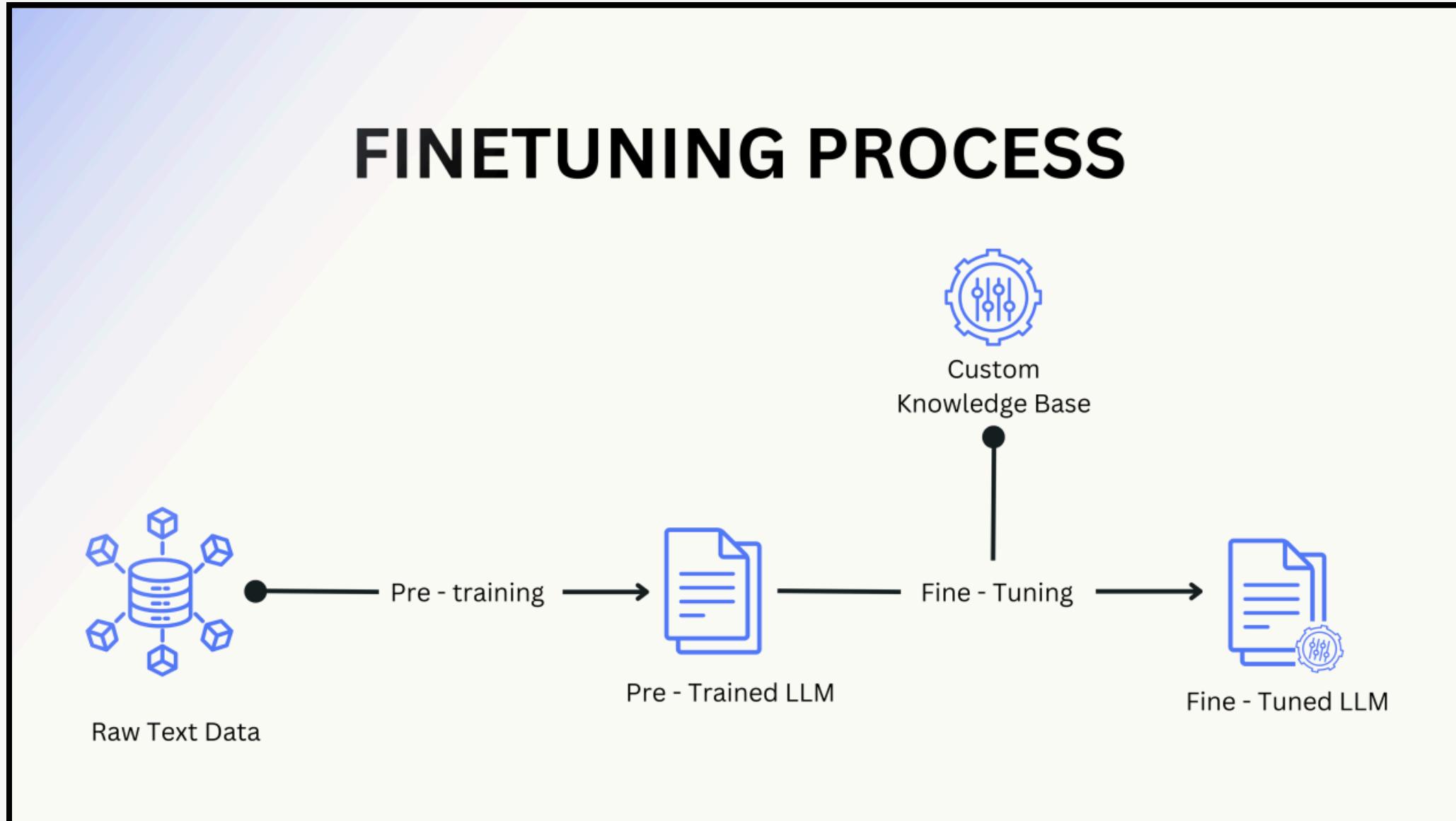
***Fine-Tuning yapmanın avantajları şunlardır:***

- LLM’i yeni bir görevi öğrenmeye zorlamak yerine, mevcut bir veri kümesinde iyi olduğu bir görevde eğiterek, LLM’in yeni görevi öğrenme olasılığı arttırılır.
- Hataya karşı dayanıklıdır. Öneriler yanlış veya eksik olsa bile, LLM’in mevcut bir veri kümesi üzerinde eğitilmesi, doğru sonuçlar üretme olasılığını artırabilir.

***Fine-Tuning yapmanın dezavantajları şunlardır:***

- Zaman alıcıdır. LLM’yi eğitmek için yeni bir veri kümesi oluşturmak veya mevcut bir veri kümescini etiketlemek gereklidir.
- Veri gereksinimi yüksektir. LLM’yi yeni bir görevi öğrenmeye zorlamak yerine, mevcut bir veri kümesinde iyi olduğu bir görevde eğitmek için daha fazla veriye ihtiyaç duyabilir.

# FINE-TUNING SÜRECİ: NASIL YAPILIR?



## SON ADIM: Modelin Test Edilmesi

Eğitilen model, daha önce görmediği bir test veri seti üzerinde performans açısından test edilmelidir. Performansın değerlendirilmesi için accuracy (doğruluk), precision, recall ve F1-score gibi metrikler kullanılır.

Görsel 41

# Fine-Tuning sırasında en çok kullanılan parametreler:

## 1. Öğrenme Oranı (Learning Rate)

- Açıklama: Ağırlık güncellemlerinin boyutunu belirler.
- Aralık: Genellikle  $1e-5$  ile  $5e-4$  arasında.

## 2. Batch Size

- Açıklama: Her adımda modele beslenen örnek sayısını ifade eder.
- Aralık: Genellikle 8, 16, 32, 64 veya 128 gibi değerler.

## 3. Epoch Sayısı

- Açıklama: Modelin eğitim verisi üzerinde kaç kez geçeceğini belirler.
- Aralık: Genelde 1 ile 10 arasında.

## 4. Ağırlık Azaltma (Weight Decay)

- Açıklama: Aşırı öğrenmeyi önlemek için ağırlıkları sınırlayan bir terimdir.
- Aralık: 0.0 ile 0.1 arasında, sıkça 0.01 veya 0.001

## 5. Sıcaklık (Temperature)

- Açıklama: Tahminlerin belirsizliğini ayarlar.
- Aralık: Genellikle 0.5 ile 2.0 arasında.

# Fine-Tuning sırasında en çok kullanılan parametreler:

## 6. İçsel Öğrenme (*Gradient Clipping*)

- Açıklama: Aşırı büyük gradyanları sınırlar.
- Aralık: Genellikle 0.5 ile 5.0 arasında.

## 7. Dropout Oranı

- Açıklama: Aşırı öğrenmeyi önlemek için belirli nöronları devre dışı bırakır.
- Aralık: Genellikle %10 ile %50 arasında (örneğin, 0.1 ile 0.5).

## 8. Optimizasyon Algoritması

- Açıklama: Öğrenme sürecini kontrol eder.
- Aralık: Bu bir parametre değildir, ancak yaygın algoritmalar arasında Adam, SGD ve RMSprop bulunur.

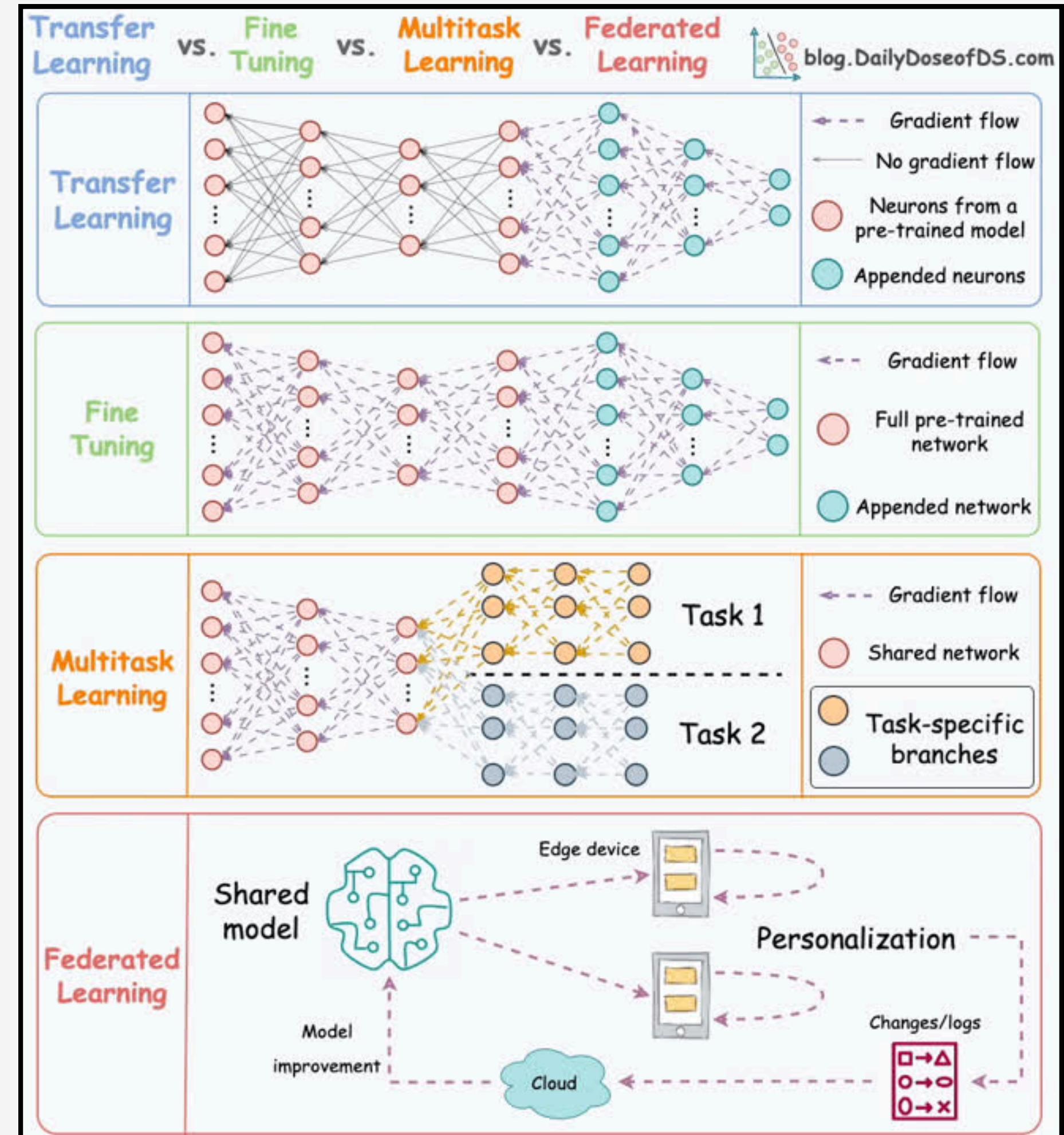
## 9. Veri Artırma (*Data Augmentation*)

- Açıklama: Eğitim veri setini genişletmek için teknikler kullanılır.
- Aralık: Belirli bir aralık yok, teknikler uygulamaya göre değişir (örneğin, kelime değiştirme, eşanlamlı kullanma).

## 10. Doğrulama Seti Boyutu (*Validation Set Size*)

- Açıklama: Model performansını değerlendirmek için ayrılan veri setinin büyüklüğünü belirtir.
- Aralık: Genellikle eğitim verisinin %5 ile %20'si arasında.

## Doğal Dil İşleme



Görsel 42

## Doğal Dil İşleme (NLP) uygulamaları genellikle iki farklı seviyede ele alınabilir:

Doğal dil işleme (NLP) uygulamalarının bir alt kümesini, **dizi düzeyi** ve **belirteç düzeyi** olarak iki ana başlıkta ele alabiliriz. Dizi düzeyinde, BERT modelinin metin girişlerini tek metin sınıflandırması, metin çifti sınıflandırması ya da metinler arasındaki bağlanımda nasıl kullanıldığını ve bu girişlerin çıktı etiketlerine nasıl dönüştürüldüğünü açıklayacağız. Belirteç düzeyinde ise, metin etiketleme ve soru yanıtlama gibi yeni uygulamalardan bahsedip, BERT'in bu tür uygulamalarda girdileri nasıl işlediğini ve çıktı etiketlerine nasıl dönüştürebileceğini detaylandıracıız. İnce ayar (fine-tuning) aşamasında, BERT'in farklı uygulamalar için ihtiyaç duyduğu "**asgari mimari değişikliklerin**" ek tam bağlı katmanlar olduğunu göreceğiz. Aşağı akış uygulamalarında bu ek katmanların parametreleri sıfırdan öğrenilirken, önceden eğitilmiş BERT modelinin tüm parametreleri ince ayar süreciyle güncellenir

## **Doğal Dil İşleme (NLP) uygulamaları genellikle iki farklı seviyede ele alınabilir:**

- 1. Dizi Düzeyi**
- 2. Belirteç Düzeyi**

Bu sunumda, BERT modelinin çeşitli NLP uygulamalarında nasıl kullanıldığını inceleyeceğiz ve girdilerin nasıl temsil edileceğine, çıktı etiketlerinin nasıl dönüştürüleceğine dair bilgi vereceğiz.

# Dizi Düzeyi NLP Uygulamaları

## [CLS]:

- Her metnin başına eklenen özel bir belirteçtir.
- Modelde sınıflandırma yapılırken kullanılır. Örneğin, duygusal analizi gibi bir görevde, modelin çıktısı bu belirteç üzerinden alınarak sınıflandırma yapılır.

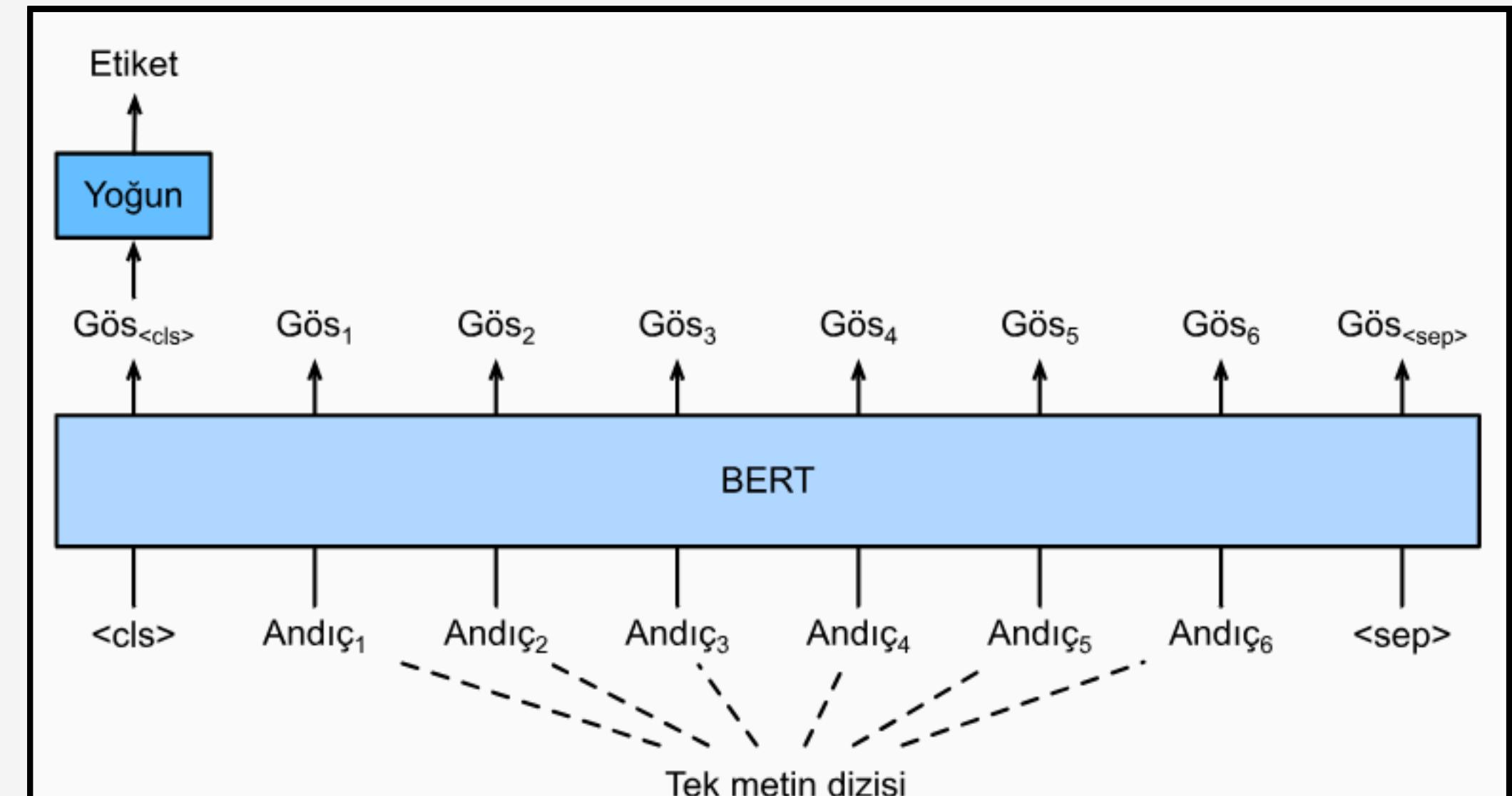
## [SEP]:

- Cümleleri veya metinleri birbirinden ayırmak için kullanılır.
- İki farklı metni işlemek ya da bir cümlenin sonunu belirtmek için eklenir. Örneğin, soru-cevap veya ilişki analizlerinde iki metni ayırır.

# Dizi Düzeyi NLP Uygulamaları

## 1. Tek Metin Sınıflandırması

- Tek bir metin parçasını ele alarak sınıflandırma işlemi yapılır.
- **Örnek Uygulama:** Duygu analizi (pozitif/negatif).
- **BERT Girdisi:** Metin dizisi, [CLS] ve [SEP] belirteçleriyle işlenir.
- **Cıktı:** Tek bir etiket – örneğin, "olumlu" veya "olumsuz".

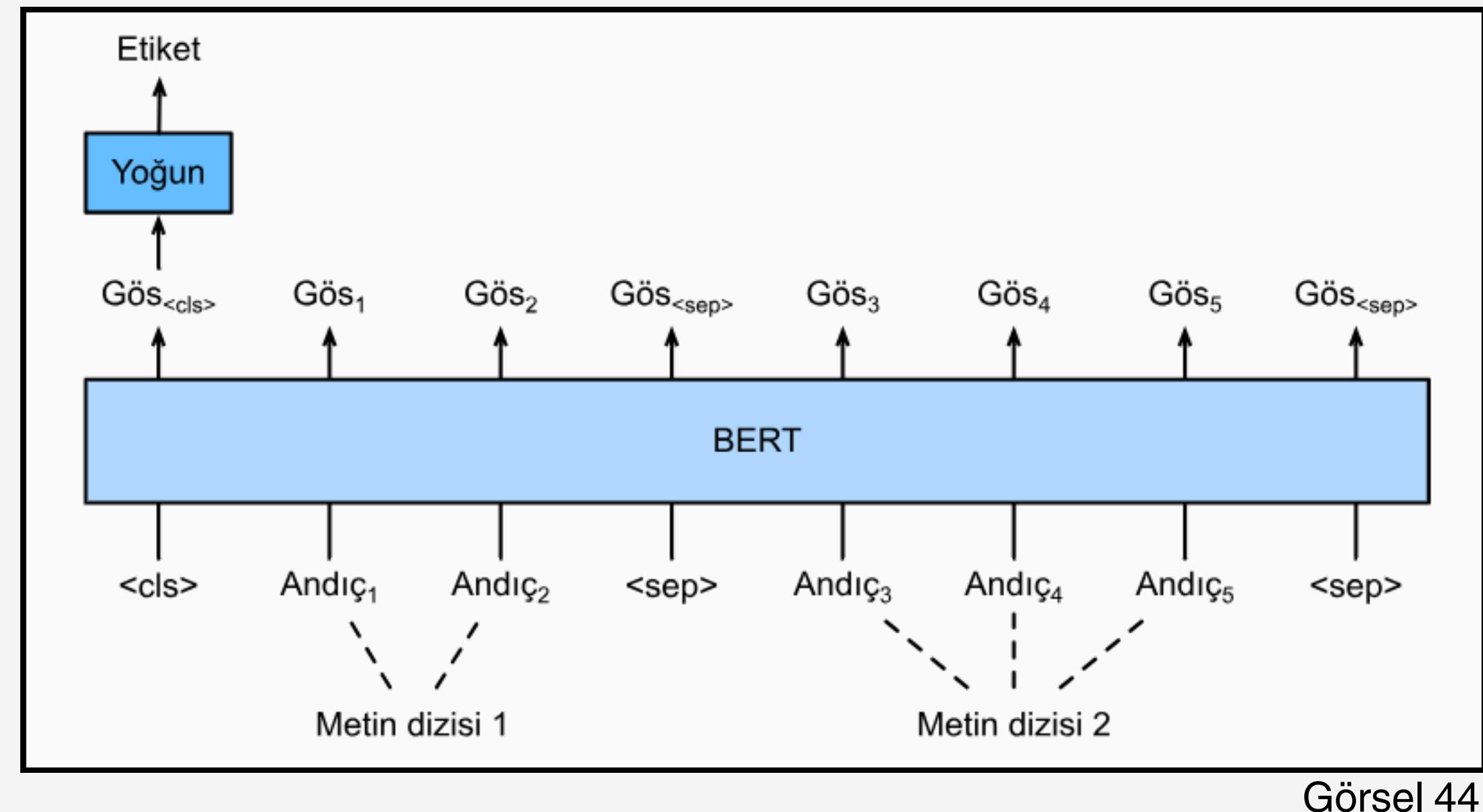


Görsel 43

# Dizi Düzeyi NLP Uygulamaları

## 2. Metin Çifti Sınıflandırması

- İki metin arasındaki ilişkiyi belirlemek amaçlanır.
- **Örnek Uygulama:** İki cümle arasındaki bağlanım (entailment).
- **BERT Girdisi:** Metin çiftleri [CLS] ve [SEP] belirteçleri ile birleştirilir.
- **Çıktı:** İkili sınıflandırma (bağlanıyor veya bağlanmıyor).



Görsel 44

# Dizi Düzeyi NLP Uygulamaları

- **0:** Cümleler arasında anlam çakışması yok,
- **5:** Cümleler eşdeğer anlamda.

Bu görevde amaç, iki cümle arasındaki benzerliği bu ölçüye göre tahmin etmektir.

**Anlamsal Metinsel Benzerlik Kıyaslama** veri kümesinden örnekler:

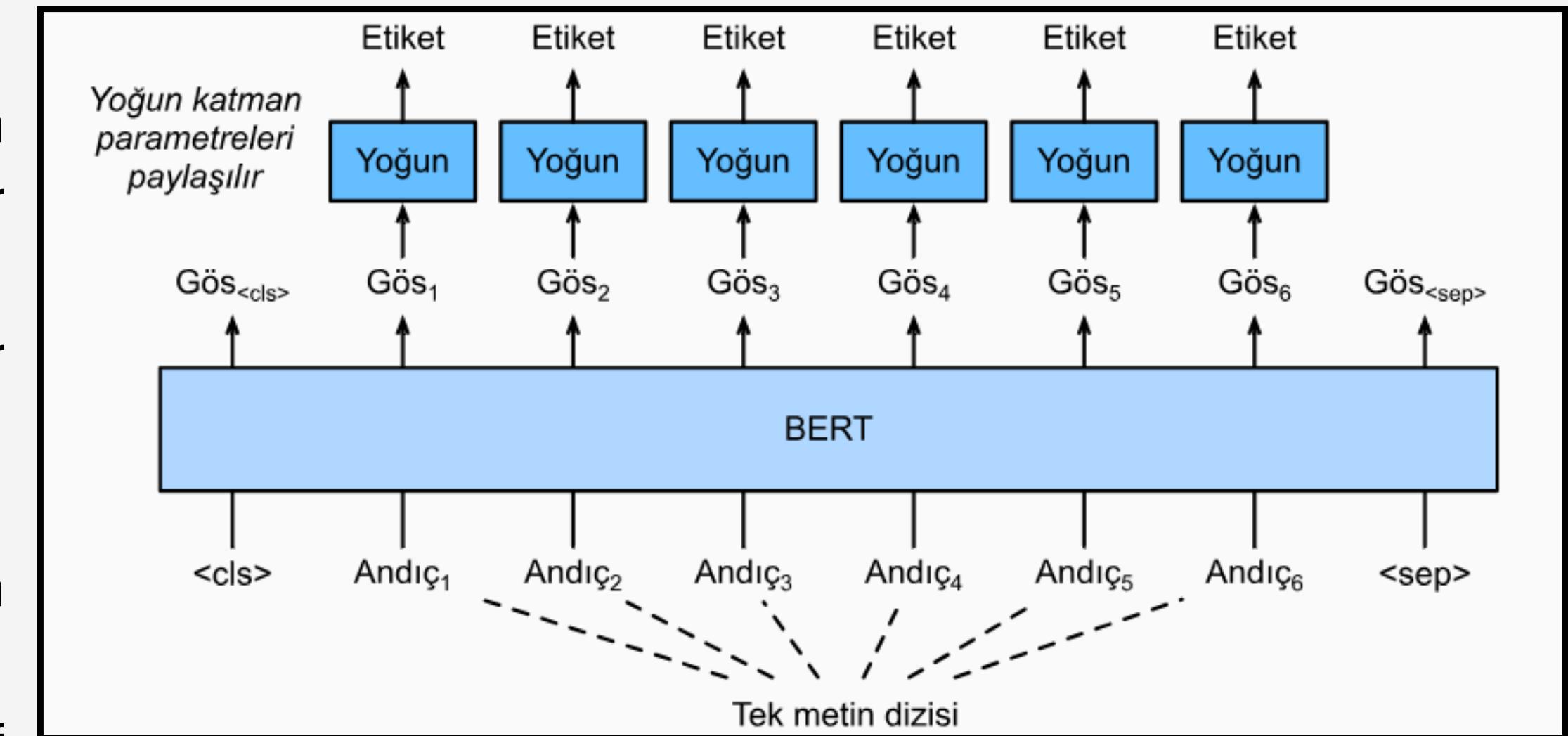
- **Cümle 1:** “Bir uçak kalkıyor.”
- **Cümle 2:** “Bir tayyare kalkıyor.”
- **Benzerlik Puanı:** 5.0 (anlamca eşdeğer)
- **Cümle 1:** “Bir kadın bir şeyler yiyor.”
- **Cümle 2:** “Bir kadın et yiyor.”
- **Benzerlik Puanı:** 3.0 (anlamca kısmen benzer)
- **Cümle 1:** “Bir kadın dans ediyor.”
- **Cümle 2:** “Bir adam konuşuyor.”
- **Benzerlik Puanı:** 0.0 (anlamca tamamen farklı)

Amaç, verilen cümle çiftlerinin anlam bakımından ne kadar benzer olduğunu tahmin etmektir.

# Belirteç Düzeyi NLP Uygulamaları

## 1. Metin Etiketleme (NER - İsim Varlık Tanıma)

- Belirteç düzeyinde her kelimenin ayrı ayrı etiketlendiği bir uygulama.
- **Örnek Uygulama:** Her bir kelimenin bir varlık sınıfına (insan, yer, organizasyon vb.) atanması.
- **BERT Girdisi:** Tek bir çümlenin belirteçleri işlenir.
- **Çıktı:** Her belirteç için bir sınıf etiketi (örn. Kişi, Lokasyon).

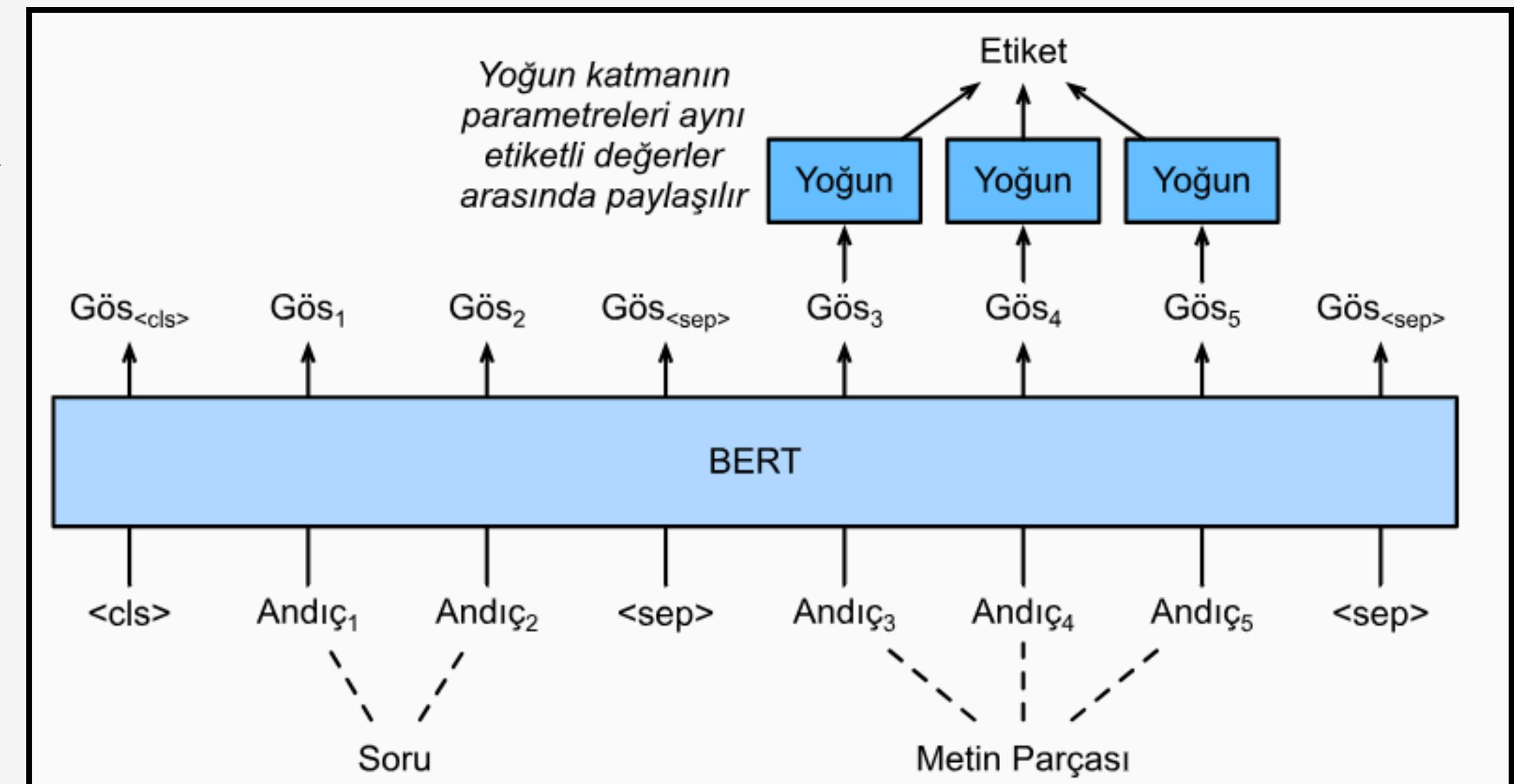


Görsel 45

# Belirteç Düzeyi NLP Uygulamaları

## 2. Soru Yanıtlama

- Bir soruya cevap verebilecek metin segmentini bulmayı amaçlar.
- **BERT Girdisi:** Soru ve pasaj birlikte [CLS] ve [SEP] belirteçleriyle kodlanır.
- **Çıktı:** Cevabın başladığı ve bittiği belirteçlerin konumları.



Görsel 46

# DİNLEDİĞİNİZ İÇİN TEŞEKKÜR EDERİZ!

Mehmet Göktuğ Gökçe		212923025
Rabia Durgut		212923003
Aslı Şemşimoğlu		212923001
Sinan Malak		212923008
Bilgehan Bayrak		212923009

# KAYNAKLAR

- [https://www.researchgate.net/publication/379513103\\_Natural\\_Language\\_Processing\\_NLP](https://www.researchgate.net/publication/379513103_Natural_Language_Processing_NLP)
- [https://www.linkedin.com/posts/allenshashaty\\_transfer-learning-vs-fine-tuning-vs-multitask-activity-7218366264237264897-2Q0O/](https://www.linkedin.com/posts/allenshashaty_transfer-learning-vs-fine-tuning-vs-multitask-activity-7218366264237264897-2Q0O/)
- [https://github.com/ayyucekizrak/Udemy\\_DerinOgrenmeyeGiris/blob/master/TransferOgrenme\\_FineTuning/ReadMe.md](https://github.com/ayyucekizrak/Udemy_DerinOgrenmeyeGiris/blob/master/TransferOgrenme_FineTuning/ReadMe.md)
- <https://furkangulsen.medium.com/tensorflow-ile-transfer-learning-i%C8%97nce-ayarlama-fine-tuning-bb1caa9d05d>
- <https://miuul.com/blog/as%C4%B1r%C4%B1-ogrenme-problemleri-nedir-ve-nas%C4%B1l-%C3%A7ozulur>
- <https://medium.com/machine-learning-t%C3%BCrkisi/attention-mekanizmasi-6d3566c0518e>
- <https://www.linkedin.com/pulse/transformer-modelleri-ve-attention-mekanizmas%C4%B1-ugur-akdogan-gevyf?originalSubdomain=tr>
- [https://medium.com/@enes\\_flz/attention-%EF%B8%8F-flash-attention-36eec4a24a1e](https://medium.com/@enes_flz/attention-%EF%B8%8F-flash-attention-36eec4a24a1e)
- <https://zeo.org/tr/ai-sozlugu/dikkat-mekanizmasi>
- <https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/>
- <https://medium.com/huawei-developers-tr/transformer-modeli-ve-%C3%A7al%C4%B1%C5%9Fma-prensipleri-bffdc4c5af0c>
- <https://blog.openzeka.com/ai/transformer-modeli-nedir/>
- <https://medium.com/@sujathamudadla1213/explain-tf-idf-in-natural-language-processing-779896d587f1> NLP'de TF-IDF'yi Anlamak: Kapsamlı Bir Kılavuz | tarafından Pradeep | Orta NLP görevleri için metin kodlama. Metinleri kodlamak en çok okunan metinlerden biridir... | tarafından DataCan | Geek Kültürü | Orta Bag of Words Model in NLP Explained | Built In
- <https://medium.com/machine-learning-t%C3%BCrkisi/nlp-ve-embedding-kullanımı-tensorflow-hub-nasıl-kullanılır-70cab0d0562d>
- <https://miuul.com/blog/encoding-nedir-turleri-nelerdir>
- [https://hasan-amanet.medium.com/n-gram-modeli-nedir-111a9053a198#:~:text=N-gram%20Modeller%20olasılık%20ve%20istatistiksel%20doğal%20dil%20işleme%20gibi,olma%20olasılığını%20tahmin%20etmeye%20çalışmaktadır.](https://hasan-amanet.medium.com/n-gram-modeli-nedir-111a9053a198#:~:text=N-gram%20Modeller%20olasılık%20ve%20istatistiksel%20doğal%20dil%20işleme%20gibi,olma%20olasılığını%20tahmin%20etmeye%20çalışmaktadır)
- <https://medium.com/@abhishekjainindore24/n-grams-in-nlp-a7c05c1aff12>
- <https://medium.com/data-science-data-engineering/time-series-prediction-lstm-bi-lstm-gru-99334fc16d75>
- [https://medium.com/@AI\\_MLwithJonny/fine-tuning-transformers-with-hugging-face-for-text-classification-a10d1b05f961](https://medium.com/@AI_MLwithJonny/fine-tuning-transformers-with-hugging-face-for-text-classification-a10d1b05f961)
- <https://medium.com/huawei-developers-tr/transformer-modeli-ve-%C3%A7al%C4%B1%C5%9Fma-prensipleri-bffdc4c5af0c>
- <https://medium.com/intel-tech/ai-dance-party-foundation-vs-fine-tuned-models-d269df518b92>
- <https://medium.com/@vishal025/decoding-lstms-6fc292a4adb0>
- <https://medium.com/@mcvarer/rnn-lstm-ve-gru-modellerinin-incelemesi-f59a73499edb>
- <https://python-bloggers.com/2023/04/text-mining-in-python-tf-idf/>
- <https://tr.d2l.ai/>

# KAYNAKLAR

- **Görsel 1:** Web, P. (2023, November 19). Web tasarım. Platform Web Reklam SEO. <https://www.webseoads.net/web-tasarim/>
- **Görsel 2:** Sarı, C. (2023, March 24). Finansal teknolojilerde chatbot'lar. FinTech İstanbul. <https://fintechistanbul.org/2023/03/27/finansal-teknolojilerde-chatbotlar/>
- **Görsel 3:** Hoş, S. (2023, February 21). Doğal Dil İşleme, NLP (Natural Language Processing) nedir? - hosting.com.tr. Doğal Dil İşleme, NLP (Natural Language Processing) Nedir? <https://www.hosting.com.tr/blog/nlp/>
- **Görsel 4:** AKCA, M. F. (2020, August 26). Nedir Bu Destek Vektör Makineleri? (Makine öğrenmesi serisi-2). Medium. <https://medium.com/deep-learning-turkiye/nedir-bu-destek-vekt%C3%B6r-makineleri-makine-%C3%B6%C4%9Frenmesi-serisi-2-94e576e4223e>
- **Görsel 5:** Krasadakis, P., Sakkopoulos, E., & Verykios, V. S. (2024, February). (PDF) a survey on challenges and advances in natural language processing with a focus on legal informatics and low-resource languages. ResearchGate. [https://www.researchgate.net/publication/378017549\\_A\\_Survey\\_on\\_Challenges\\_and\\_Advances\\_in\\_Natural\\_Language\\_Processing\\_with\\_a\\_Focus\\_on\\_Legal\\_Informatics\\_and\\_Low-Resource\\_Languages](https://www.researchgate.net/publication/378017549_A_Survey_on_Challenges_and_Advances_in_Natural_Language_Processing_with_a_Focus_on_Legal_Informatics_and_Low-Resource_Languages)
- **Görsel 6:** Beden, Ö. (2019, October 28). Derin öğrenme – 1 – ileri Yayılm Algoritması. Yapay Akademi. <https://yapayakademi.com/derin-ogrenme-1-ileri-yayilim-algoritmasi/>
- **Görsel 7:** Obasi, F. (2024, May 20). How Bert simplifies NLP with text encoder representations: Franklin Obasi posted on the topic. LinkedIn. [https://www.linkedin.com/posts/franklinobasy\\_nlp-artificialintelligence-machinelearning-activity-7198348869405966336-mhcB/](https://www.linkedin.com/posts/franklinobasy_nlp-artificialintelligence-machinelearning-activity-7198348869405966336-mhcB/)
- **Görsel 8:** Novita.ai. (2024, April 22). What are large language models (llms)? Novita AI. <https://blogs.novita.ai/what-are-large-language-models-llms/>
- **Görsel 9:** Duygu Analizi. ProxyElite. (n.d.). <https://proxyelite.info/tr/glossary/sentiment-analysis/>
- **Görsel 10:** Durna, M. B. (2024, January 10). Text representation techniques. Medium. <https://medium.com/@mervebdurna/text-representation-techniques-d40741eb0916>
- **Görsel 11:** Aksu, H. (2024, May 2). Problem - "Metinlerdeki Değerli İpuçlarını Ortaya Çıkarmak: TF-IDF'nin Rolü." GitLab. [https://gitlab.bulutbilisimciler.com/bb-public/scenarios/-/blob/0a703f012b4f0b7799dc6cb58a6aee15d5b905fb/2024-05-02/problem-2024-05-02/tr\\_step1.md](https://gitlab.bulutbilisimciler.com/bb-public/scenarios/-/blob/0a703f012b4f0b7799dc6cb58a6aee15d5b905fb/2024-05-02/problem-2024-05-02/tr_step1.md)
- **Görsel 12:** Bostancı, V. M. (2023, September 3). TF-IDF. LinkedIn. <https://www.linkedin.com/pulse/tf-idf-volkan-musa-bostanci/>
- **Görsel 13:** Techslang. (2022, September 2). What is encoding? - definition by Techslang. <https://www.techslang.com/definition/what-is-encoding/>
- **Tablo 1:** Tafralı, S. (2022, May 11). Veri Biliminde Encoding İşlemleri. Medium. <https://serdartafrali.medium.com/veri-biliminde-encoding-i%C3%87%C5%9Flemleri-616e87bf8c74>
- **Görsel 14:** Analı, S. (2024, February 15). A Guide to Word Embedding. Medium. <https://towardsdatascience.com/a-guide-to-word-embeddings-8a23817ab60f>
- **Görsel 15:** Bıçakçı, K. (2021, October 21). NLP ve Embedding I TensorFlow Hub Nedir?. Medium. <https://medium.com/machine-learning-t%C3%BCrkiye/nlp-ve-embedding-kullan%C4%B1m%C4%B1-tensorflow-hub-nas%C4%B1l-kullan%C4%B1%C4%B1r-70cab0d0562d> Görsel 16: Zhou, M., Duan, N., Liu, S., & Shum, H. Y. (2020, January). Progress in Neural NLP: Modeling, Learning, and Reasoning. ResearchGate. [https://www.researchgate.net/publication/338426630\\_Progress\\_in\\_Neural\\_NLP\\_Modeling\\_Learning\\_and\\_Reasoning](https://www.researchgate.net/publication/338426630_Progress_in_Neural_NLP_Modeling_Learning_and_Reasoning) Görsel 17: Şebin, B. (2022, June 29). Word2vec ve Glove — Türkçe NLP. Medium. <https://medium.com/%40sebinbusra/word2vec-ve-glove-t%C3%BCrkiye-nlp-164aa7443a0f>
- **Görsel 18:** Şebin, B. (2022, June 29). Word2vec ve Glove — Türkçe NLP. Medium. <https://medium.com/%40sebinbusra/word2vec-ve-glove-t%C3%BCrkiye-nlp-164aa7443a0f>
- **Tablo 1:** Tafralı, S. (2022, May 11). Veri Biliminde Encoding İşlemleri. Medium. <https://serdartafrali.medium.com/veri-biliminde-encoding-i%C3%87%C5%9Flemleri-616e87bf8c74>
- **Görsel 19:** UniverAI. (2024, March 30). Mechanism Attention, Networks with Attention. Habr. <https://habr.com/en/articles/804079/>
- **Görsel 20:** DiveIntoDeepLearning. (n.d.). Self-Attention and Positional Encoding. Dive into Deep Learning 0.17.6 documentation. [https://classic.d2l.ai/chapter\\_attention-mechanisms/self-attention-and-positional-encoding.html](https://classic.d2l.ai/chapter_attention-mechanisms/self-attention-and-positional-encoding.html)
- **Görsel 21, Görsel 30, Görsel 31, Görsel 32, Görsel 33:** Bıçakçı, K. (2022, January 17). Transformer Encoder Yapısı I Self-Attention. Medium. <https://medium.com/machine-learning-t%C3%BCrkiye/transformer-encoder-yap%C4%B1%C4%B1s%C4%B1-self-attention-c44564d0b74b>
- **Görsel 24:** Singh, K. (2024, April 22). Building Blocks for AI Part 4: Types of Neural Networks. Medium. <https://medium.com/@kamalmeet/building-blocks-for-ai-part-4-types-of-neural-networks-1fe1ed234132>
- **Görsel 25:** DiveIntoDeepLearning. (n.d.-c). Uzun Ömürlü Kısa-Dönem Belleği (LSTM). Dive into Deep Learning 0.17.6 documentation. [https://tr.d2l.ai/chapter\\_recurrent-modern/lstm.html](https://tr.d2l.ai/chapter_recurrent-modern/lstm.html)
- **Görsel 26:** Bezza, F. Z. (2024, June 26). Time Series Prediction: LSTM ,BI-LSTM ,GRU. Medium. <https://medium.com/data-science-data-engineering/time-series-prediction-lstm-bi-lstm-gru-99334fc16d75>
- **Görsel 27:** Çayırlı, H. (2024, January 15). Transformer Modeli ve Çalışma Prensipleri. Medium. <https://medium.com/huawei-developers-tr/transformer-modeli-ve-%C3%A7al%C4%B1%C5%9Fma-prensipleri-bffdc4c5af0c>
- **Görsel 28:** DiveIntoDeepLearning. (n.d.-c). The Transformer Architecture. Dive into Deep Learning 1.0.3 documentation. [https://d2l.ai/chapter\\_attention-mechanisms-and-transformers/transformer.html](https://d2l.ai/chapter_attention-mechanisms-and-transformers/transformer.html)
- **Görsel 29:** Bıçakçı, K. (2022a, January 17). Transformer Encoder Yapısı I Self-Attention. Medium. <https://medium.com/machine-learning-t%C3%BCrkiye/transformer-encoder-yap%C4%B1%C4%B1s%C4%B1-self-attention-c44564d0b74b>
- **Görsel 34:** Tunalı, M. M. (2020, January 27). Bölüm 2: Deep Learning Training GUI — Yeni Özellikler. Medium. <https://medium.com/deep-learning-turkiye/deep-learning-training-gui-yeni-ozellikler-22a63c8bb243>
- **Görsel 35:** Bengio, Y. (2009). Introduction to Deep Learning. <https://www.cse.iitk.ac.in/users/sigml/lec/Slides/Ram.pdf>
- **Görsel 36:** dasarpAI. (2023, August 18). What is LLM. <https://dasarpai.com/dsblog/what-is-lm>
- **Görsel 37:** Görsel 38: Kızrak, A. (2019). Fine-Tuning ve Transfer Öğrenme Nedir?. GitHub. [https://github.com/ayyucekizrak/Udemy\\_DerinOgrenmeyeGiris/blob/master/TransferOgrenme\\_FineTuning/ReadMe.md](https://github.com/ayyucekizrak/Udemy_DerinOgrenmeyeGiris/blob/master/TransferOgrenme_FineTuning/ReadMe.md)
- **Görsel 39:** Aydin, K. (2023, December 2). Everything to learn about Large Language Models(Part 2/3 Pre-training and Fine-tuning). Medium. <https://medium.com/@aydinKerem/everything-to-learn-about-large-language-models-part-2-3-pre-training-and-fine-tuning-5fc68700701a>
- **Görsel 40:** Tafralı, S. (2023, July 28). Aşırı Öğrenme Problemleri Nedir ve Nasıl Çözülür?. Miuul. <https://miuul.com/blog/as%C4%B1r%C4%B1%C4%9F-B1-ogrenme-problemleri-nedir-ve-nas%C4%B1r%C4%B1-%C3%A7ozulur>
- **Görsel 41:** Tagx. (2024, June 20). What is Supervised Fine-Tuning?. LinkedIn. <https://www.linkedin.com/pulse/what-supervised-fine-tuning-tagx-yq8if/>
- **Görsel 42:** Community, B. (2023, December 19). An overview of transfer learning vs fine tuning vs multitask learning vs federated learning!. LinkedIn. [https://www.linkedin.com/posts/brain-community\\_transferlearning-finetuning-federatedlearning-activity-7142876261399330817-Ys0d/](https://www.linkedin.com/posts/brain-community_transferlearning-finetuning-federatedlearning-activity-7142876261399330817-Ys0d/)
- **Görsel 43: Görsel 44,Görsel 45,Görsel 46:** DiveIntoDeepLearning. (n.d.-b). Dizi Düzeyinde ve Belirteç Düzeyinde Uygulamalar için BERT İnce Ayarı. Derin Öğrenmeye Daliş 0.17.5 documentation. [https://tr.d2l.ai/chapter\\_natural-language-processing-applications/finetuning-bert.html](https://tr.d2l.ai/chapter_natural-language-processing-applications/finetuning-bert.html)