



T.C. SANAYİ VE  
TEKNOLOJİ BAKANLIĞI



# BÜYÜK DİL MODELLERİ VE İSTEM MÜHENDİSLİĞİ

ARAŞTIRMA SERİSİ - SAYI 24

```
// Game logic simulation
function gameLoop() {
    console.log("Game starting...");  

    const player = players.find(p => p.id === playerId);
    if (!player) return;  

    const lives = player.lives;
    const boardSize = GAME_SETTINGS.boardSize;
    const maxLives = GAME_SETTINGS.initialLives;  

    const diceRoll = rollDice();
    console.log(`Player ${player.name} rolled a ${diceRoll}.`);  

    if (diceRoll === 0) {
        console.error(`Player ${player.name} rolled a 0!`);  

        if (player.lives <= 0) {
            console.log(`Player ${player.name} has reached the final line!`);  

            return;
        }
        player.position -= boardSize;
        console.log(`Player ${player.name} moved back by ${boardSize} cells`);  

        if (player.position <= 0) {
            console.error(`Player ${player.name} is out of the game!`);  

            return;
        }
        player.lives--;
        console.log(`Player ${player.name} lost a life. Remaining lives: ${player.lives}`);  

    } else if (diceRoll === 1) {
        console.log(`Player ${player.name} rolled a 1!`);  

        if (player.position + boardSize > boardSize) {
            console.error(`Player ${player.name} rolled a ${diceRoll}, but it's a wrap!`);  

            return;
        }
        player.position += boardSize;
        console.log(`Player ${player.name} moved forward by ${boardSize} cells`);  

        if (player.position === boardSize) {
            console.log(`Player ${player.name} has reached the finish line!`);  

            return;
        }
        player.lives--;
        console.log(`Player ${player.name} lost a life. Remaining lives: ${player.lives}`);  

    } else if (diceRoll === 2) {
        console.log(`Player ${player.name} rolled a 2!`);  

        if (player.position + boardSize * 2 > boardSize) {
            console.error(`Player ${player.name} rolled a ${diceRoll}, but it's a wrap!`);  

            return;
        }
        player.position += boardSize * 2;
        console.log(`Player ${player.name} moved forward by ${boardSize * 2} cells`);  

        if (player.position === boardSize * 2) {
            console.log(`Player ${player.name} has reached the finish line!`);  

            return;
        }
        player.lives--;
        console.log(`Player ${player.name} lost a life. Remaining lives: ${player.lives}`);  

    } else if (diceRoll === 3) {
        console.log(`Player ${player.name} rolled a 3!`);  

        if (player.position + boardSize * 3 > boardSize) {
            console.error(`Player ${player.name} rolled a ${diceRoll}, but it's a wrap!`);  

            return;
        }
        player.position += boardSize * 3;
        console.log(`Player ${player.name} moved forward by ${boardSize * 3} cells`);  

        if (player.position === boardSize * 3) {
            console.log(`Player ${player.name} has reached the finish line!`);  

            return;
        }
        player.lives--;
        console.log(`Player ${player.name} lost a life. Remaining lives: ${player.lives}`);  

    } else if (diceRoll === 4) {
        console.log(`Player ${player.name} rolled a 4!`);  

        if (player.position + boardSize * 4 > boardSize) {
            console.error(`Player ${player.name} rolled a ${diceRoll}, but it's a wrap!`);  

            return;
        }
        player.position += boardSize * 4;
        console.log(`Player ${player.name} moved forward by ${boardSize * 4} cells`);  

        if (player.position === boardSize * 4) {
            console.log(`Player ${player.name} has reached the finish line!`);  

            return;
        }
        player.lives--;
        console.log(`Player ${player.name} lost a life. Remaining lives: ${player.lives}`);  

    } else if (diceRoll === 5) {
        console.log(`Player ${player.name} rolled a 5!`);  

        if (player.position + boardSize * 5 > boardSize) {
            console.error(`Player ${player.name} rolled a ${diceRoll}, but it's a wrap!`);  

            return;
        }
        player.position += boardSize * 5;
        console.log(`Player ${player.name} moved forward by ${boardSize * 5} cells`);  

        if (player.position === boardSize * 5) {
            console.log(`Player ${player.name} has reached the finish line!`);  

            return;
        }
        player.lives--;
        console.log(`Player ${player.name} lost a life. Remaining lives: ${player.lives}`);  

    } else if (diceRoll === 6) {
        console.log(`Player ${player.name} rolled a 6!`);  

        if (player.position + boardSize * 6 > boardSize) {
            console.error(`Player ${player.name} rolled a ${diceRoll}, but it's a wrap!`);  

            return;
        }
        player.position += boardSize * 6;
        console.log(`Player ${player.name} moved forward by ${boardSize * 6} cells`);  

        if (player.position === boardSize * 6) {
            console.log(`Player ${player.name} has reached the finish line!`);  

            return;
        }
        player.lives--;
        console.log(`Player ${player.name} lost a life. Remaining lives: ${player.lives}`);  

    }
}
```

BİLGEM

YAZILIM TEKNOLOJİLERİ ARAŞTIRMA ENSTİTÜSÜ

# Simge ve Kısaltmalar

Kısaltmalar	Açıklama
TÜBİTAK	Türkiye Bilimsel ve Teknolojik Araştırma Kurumu
BİLGEM	Bilişim ve Bilgi Güvenliği İleri Teknolojiler Araştırma Merkezi
YTE	Yazılım Teknolojileri Araştırma Enstitüsü
BART	Bidirectional and Auto-Regressive Transformer (Çift Yönlü ve Oto-Regresif Dönüştürücü)
BERT	Bidirectional Encoder Representations from Transformers (Çift Yönlü Kodlayıcı Temsilleri)
CoT	Chain of Thought (Düşünce Zinciri)
CoT SC	Chain of Thought - Self-Consistency (Kendine Tutarlı Düşünce Zinciri)
GoT	Graph of Thoughts (Düşünce Grafiği)
GPT	Generative Pretrained Transformer (Üretken Ön İşlemeli Dönüştürücü)
GRU	Gated Recurrent Unit (Geçitli Tekrarlayan Ünite)
LLM	Large Language Model (Büyük Dil Modeli)
LSTM	Long Short-Term Memory (Uzun Kısa Süreli Bellek)
MLM	Masked Language Model (Maskelenmiş Dil Modeli)
MoE	Mixture of Experts (Uzman Karışımlı)
NLP	Natural Language Processing (Doğal Dil İşleme)
NSP	Next Sentence Prediction (Sonraki Cümle Tahmini)
PoT	Program of Thoughts (Düşünce Programı)
RNN	Recurrent Neural Network (Özyinelemeli Sinir Ağı)
SLM	Small Language Model (Küçük Dil Modeli)
SoTA	State-of-the-Art (En Güncel Teknoloji)
T5	Text-to-Text Transfer Transformer (Metinden Metne Dönüşüm Modeli)
ToT	Tree of Thoughts (Düşünce Ağacı)

**Yazar**

Nezahat KORKMAZ

**Yayın Koordinatörü**

Beyza ŞENEL

**Editörler**

Eyüp Halit YILMAZ

Mehmet Emin KÜÇÜKER

Muhammet Furkan ÇOŞKUN

Kübra ERTÜRK

Nurhan ÖNER

Tuğçe YILMAZ

**Tasarım**

Özge DOĞAN

**©2025 - Tüm hakları saklıdır.**

**İletişim:** 0(312) 289 92 22 - [YTE.Bilgi@tubitak.gov.tr](mailto:YTE.Bilgi@tubitak.gov.tr)

<https://bilgem.tubitak.gov.tr/YTE/>

Yayımlanan yazıların sorumluluğu yazarlarına aittir, TÜBİTAK BİLGEM sorumlu tutulamaz.

# İçindekiler

---

Ön Söz	5
Giriş	6
Büyük Dil Modelleri ve Gelişimi	7
Büyük Dil Modellerinin Kullanım Alanları	16
Büyük Dil Modellerinin Güçlü ve Zayıf Yönleri	17
İstem Mühendisliğinin Tanımı ve Önemi	18
Etkili İstem Yazım Teknikleri	19
Gelişmiş İstem Teknikleri	21
Örneklerle İstemleme Yöntemleri	25
Gelecekteki Araştırma Yönetimleri	38
Sonuç ve Öneriler	39
Kaynakça	40

# Ön Söz

---

TÜBİTAK BİLGE Yazılım Teknolojileri Araştırma Enstitüsü (YTE), 2012 yılından bu yana yazılım teknolojilerinde Ar-Ge faaliyetleri yürüten bir araştırma kuruluşudur. Araştırma faaliyetlerinde elde ettiği birikimini stratejik, hassas ve kritik projeler yürüterek kamu adına hayatı geçirmekte; kurumlarımıza dijital dönüşüm, yazılım geliştirme teknolojileri ve kalite süreçleri konusunda danışmanlık vermektedir.

TÜBİTAK BİLGE YTE tarafından hazırlanan Araştırma Serisi ile kurum içi içerik üretme çalışmalarının yaygınlaştırılması ve hazırlanan içeriklerin sektörün erişimine açılması amaçlanmaktadır. Araştırma Serisi'nde yayınlanan çalışmalar TÜBİTAK BİLGE YTE çalışanlarının projelerde elde ettiği bilgi birikimini paylaşmak adına derlenmiştir. Bu çalışmalar ile ülkemizin yazılım sektörüne katkı sağlanması hedeflenmektedir.

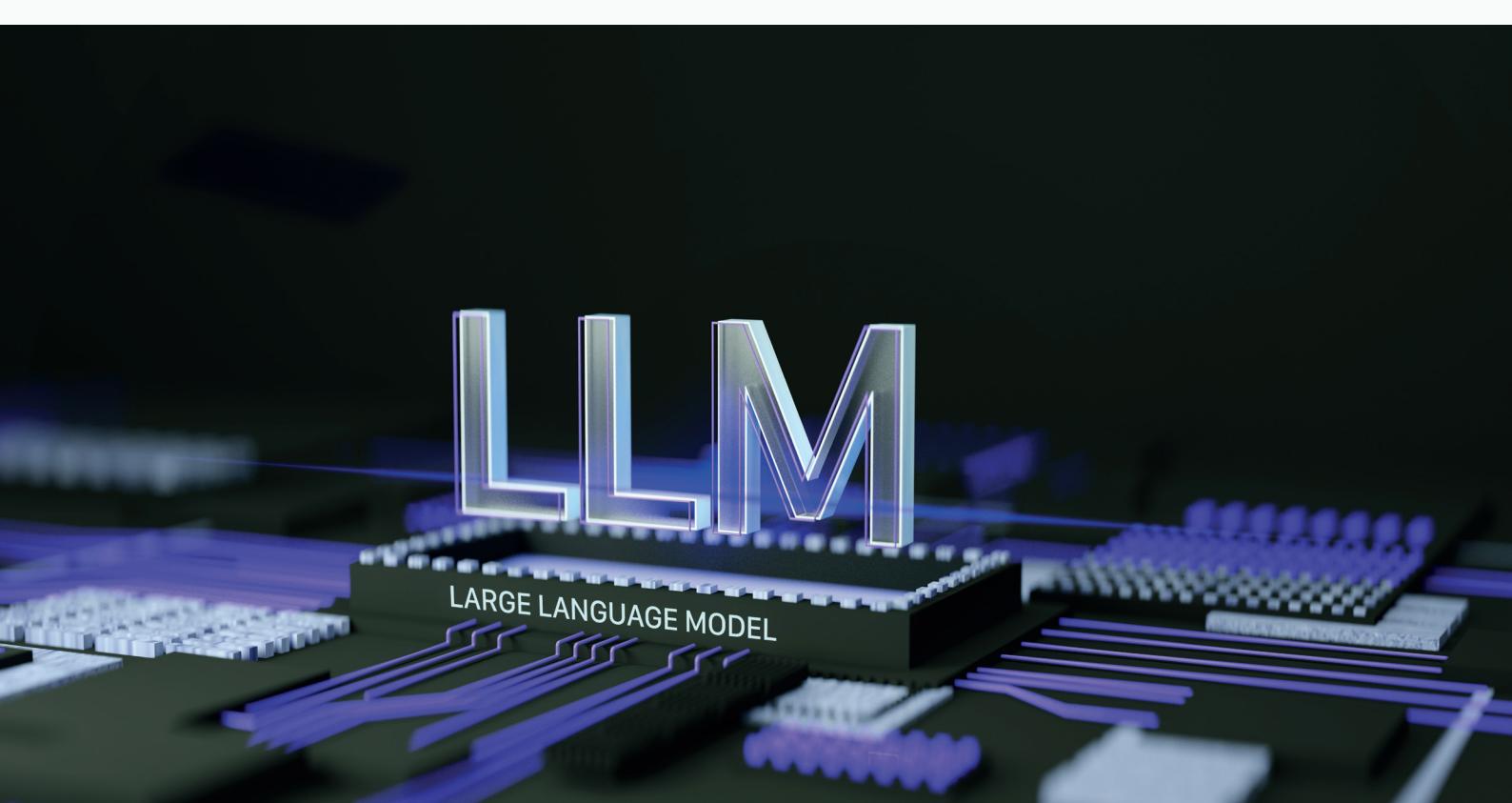
# Giriş

Yapay zeka hakkında son yıllarda kaydedilen gelişmeler, çığır açıcı yeniliklere zemin hazırlamıştır. Bu yeniliklerin başında, büyük dil modelleri [Large Language Models (LLM)] gelmektedir. Büyük dil modelleri metin üretimi, metin anlaması ve analiz etme süreçlerinde insan benzeri performans sergileyerek karmaşık dil işleme görevlerinde kolaylık sağlamaktadır. Ancak, bu modellerin sunduğu etkileyici performans, yalnızca modelin büyülüğu ve hesaplama kapasitesiyle sınırlı olmayıp, doğru ve etkili istemlerin (prompt'ların) oluşturulmasıyla yakından ilişkilidir.

Büyük dil modelleri, kelimelerin anımlarını doğrudan kavrayabilen sistemler değil; daha çok istatistiksel örüntülere dayalı yapay zeka algoritmalarıdır. Bu nedenle, kullanıcıların girdi bağlamını göz önünde bulundurarak net ve anlaşılır istemler tasarlaması büyük önem taşımaktadır.

Bu kapsamda, kullanıcılarla önemli bir pay düşmektedir. Çözülmlesi amaçlanan probleme uygun, net ve açıklayıcı istemler sağlanmalıdır. Bu da istem (prompt) mühendisliği kavramını ortaya çıkarmıştır. İstem mühendisliği, büyük dil modellerinden en verimli çıktıları almak için istem tasarlama süreçlerini iyileştiren stratejik bir yaklaşımdır. Dil modeli ve kullanıcı arasındaki etkileşimi güçlendiren bu mühendislik disiplini, istemlerin içeriğini ve bağlamını iyileştirerek daha doğru, tutarlı ve anlamlı model çıktıları elde edilmesini sağlar. Instruction tuning ve prompt tuning gibi teknikler, büyük dil modellerinin istenilen görevler için daha hassas ve hedef odaklı çalışmasını sağlayarak model çıktılarının optimizasyonuna katkıda bulunmaktadır.

Bu çalışma, büyük dil modellerinin yapısı, etkin kullanımı ve istem mühendisliği kavramını detaylı bir şekilde ele alarak, temel tekniklerden ileri düzey yöntemlere kadar geniş bir perspektif sunmayı amaçlamaktadır. Özellikle metin tamamlama, soru-cevap sistemleri ve metin özetleme gibi uygulama alanlarında doğruluk ve tutarlılığı artırmaya yönelik stratejilere odaklanılmaktadır. Ayrıca, büyük dil modellerinin sınırları, etik sorumlulukları ve gelecekteki gelişim alanları tartışılarak kapsamlı bir değerlendirme sunulmaktadır.

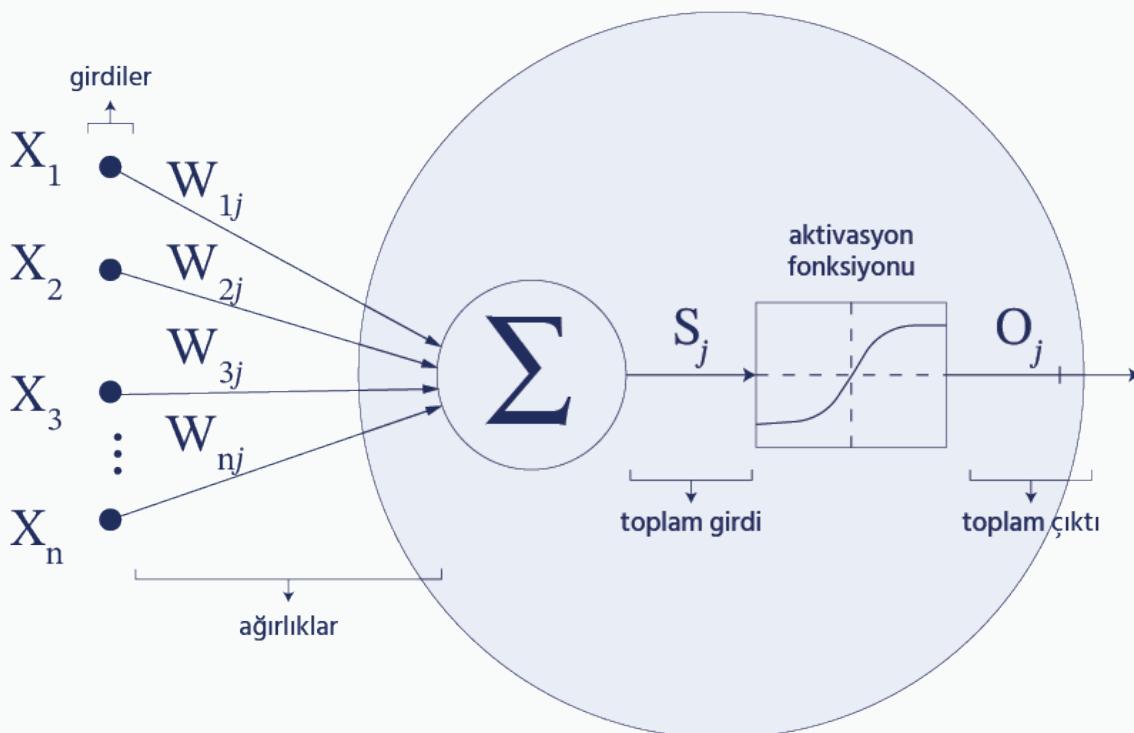


# Büyük Dil Modelleri ve Gelişimi

Büyük Dil Modelleri, doğal dili işleyerek metin üretme, anlama ve analiz etme süreçlerinde insan benzeri sonuçlar sunabilen modellerdir. Bu modeller, büyük veri kümeleri üzerinde eğitilmiş çok katmanlı derin öğrenme mimarilerine dayanır ve özellikle self-supervised (kendiliğinden denetimli) öğrenme teknikleriyle denetimli öğrenmenin getirdiği etiketleme maliyetlerini azaltarak geniş bir bilgi tabanı oluşturur [1]. Bu kapsamda istem mühendisliğini anlamlandırmak için önce yapay zeka modellerinin yapısı ve gelişimi incelenmektedir.

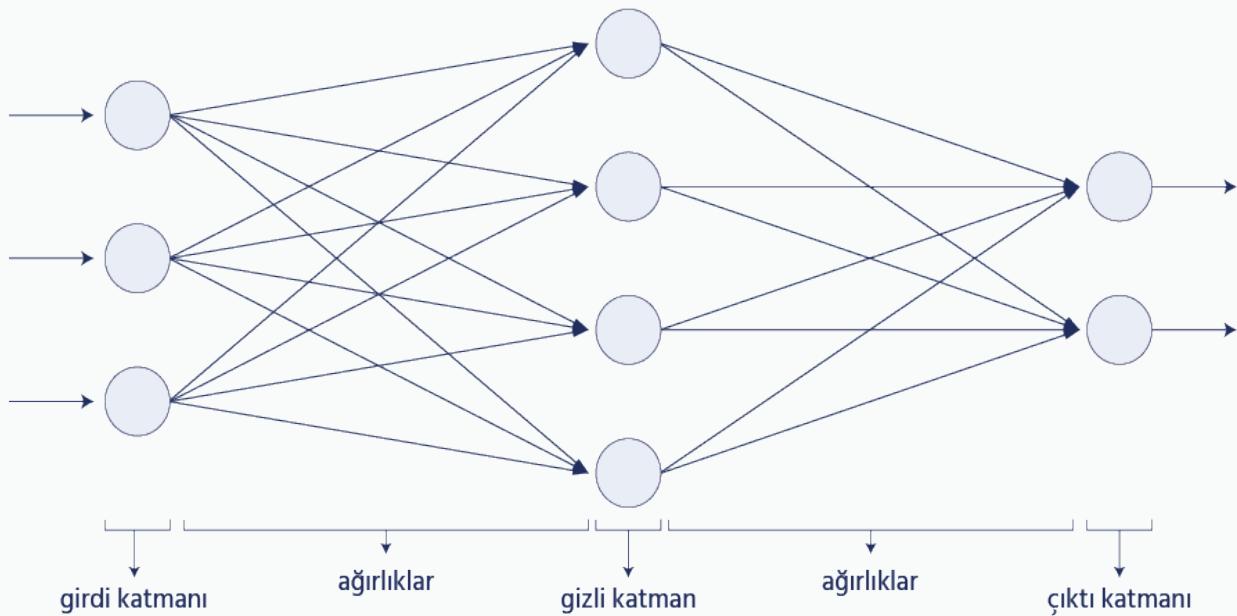
## Yapay Zeka Modellerinin Temel Yapısı

Büyük Dil Modellerinin temelini oluşturan yapay sinir ağları, birbirine bağlı yapay nöronlardan oluşur ve bu nöronlar, veriyi katmanlar boyunca işleyerek çıktılar üreten bir yapıya sahiptir. Nöron, belirli bir giriş sinyalini alır, bunu bir ağırlık değeri ile çarpar ve aktivasyon fonksiyonu ile işledikten sonra bir çıktı üretir [2].



Şekil 1. Bir Yapay Nöron Örneği [3]

Şekil 1'de gösterilen perceptron modeli (yapay nöron modeli), ilk olarak Rosenblatt tarafından tanıtılmıştır [3]. Bu basit yapı, çok katmanlı hale getirildiğinde karmaşık veri örüntülerini öğrenebilmektedir. Sinir ağlarındaki nöron katmanlarının derinliği arttıkça modelin öğrenme kapasitesi de artmaktadır. Şekil 2'de gösterildiği gibi, çok katmanlı sinir ağları mimarisini bu öğrenme kapasitesini artırmak için ek gizli katmanlar içermektedir [2].



**Şekil 2. Çok Katmanlı Sinir Ağları Mimarisi [2]**

## Yapay Zeka Modellerinin Evrimi

Yapay zeka modelleri, zamanla farklı yöntemler ve mimari değişikliklerle gelişerek bugünkü halini almıştır. İlk dönemlerde sinir ağları tabanlı yaklaşımlar benimsenmiş, ancak bu sistemler veri işleme süreçlerinde bazı sınırlamalarla karşılaşmıştır. Bu eksiklikleri gidermek amacıyla farklı modelleme teknikleri geliştirilmiş ve dil işleme süreçlerinde önemli ilerlemeler kaydedilmiştir. Bu bölümde, Özyinelemeli Sinir Ağları [Recurrent Neural Network (RNN)], Uzun Kısa Süreli Bellek [Long Short-Term Memory (LSTM)] ve Geçitli Tekrarlayan Ünite [Gated Recurrent Unit (GRU)] gibi yöntemlerden Transformer mimarisine kadar uzanan yapay zeka modellerinin evrimsel süreci ele alınarak, bu gelişim sürecinin getirdiği yenilikler incelenecaktır.

## Neural Network ve RNN Tabanlı Mimariler

İlk doğal dil işleme modelleri, 1960'lı yıllarda sinir ağlarının temel prensiplerine dayanarak geliştirilmiştir. Ancak geleneksel sinir ağları, ardisık veri işlemede yeterince etkili olamamıştır. Bu eksikliği gidermek için 1980'li yılların ortalarında Özyinelemeli Sinir Ağrı [Recurrent Neural Network (RNN)] geliştirilmiştir [5]. RNN'ler, veri dizilerini ardisık olarak işleyerek zaman-bağımlı bilgiyi kullanabilen yapılar sunmuştur. Ancak, uzun veri dizilerinde dizinin başındaki bilgilerin zamanla unutulması nedeniyle performans sorunları ortaya çıkmıştır. Bu durum, RNN'lerin geri yayılım sırasında gradyan büyükliklerinin hızla sıfıra yaklaşması (vanishing gradient) veya aşırı büyümesi (exploding gradient) nedeniyle öğrenmede zorluk yaşammasına neden olmuştur [6]. Bu sorunları çözmek için Uzun Kısa Süreli Bellek [Long Short-Term Memory (LSTM)] (1997) ve Geçitli Tekrarlayan Ünite [Gated Recurrent Unit (GRU)] (2014) gibi modeller geliştirilmiştir. Bu yaklaşımlar, kaybolan gradyan sorununu büyük ölçüde azaltarak daha verimli öğrenme sağlamıştır ([7], [8]). Ardından, 2017'de geliştirilen Transformer mimarisini, RNN'lerin uzun sekansları işleme konusundaki sınırlamalarını büyük oranda çözerek, günümüzdeki büyük dil modellerinin temelini oluşturmuştur [9].

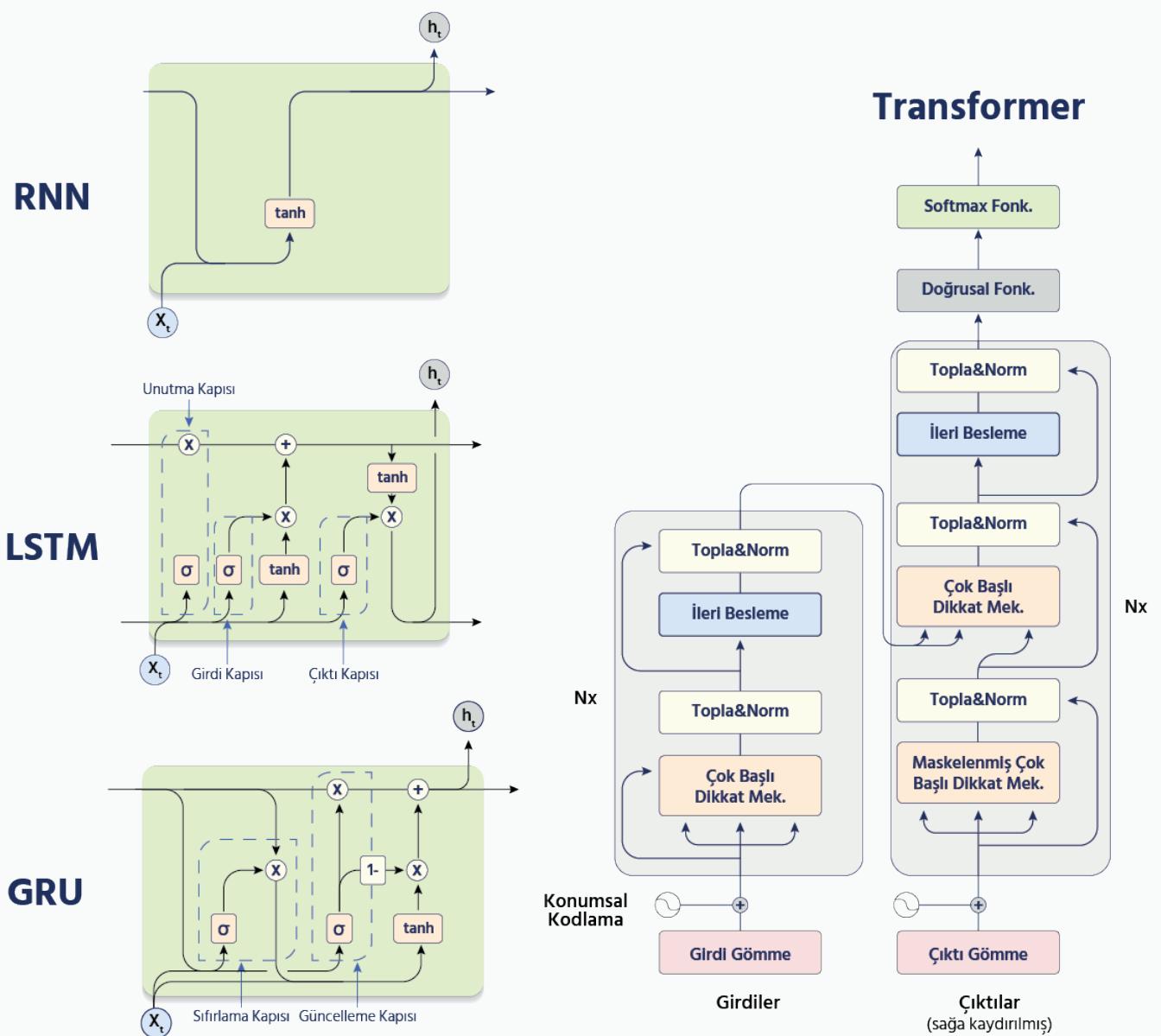
## LSTM ve GRU: RNN Geliştirmeleri

RNN'ler, uzun veri dizilerinde unutma ve bozulma gibi sorunlarla karşılaşıldığı için LSTM ve GRU modelleri geliştirilmiştir [6].

LSTM, giriş verilerini hatırlamak için hücre durumu ve kapı mekanizmaları kullanırken, GRU daha sade bir yapıya sahiptir ve hesaplama açısından daha verimlidir.

LSTM ve GRU, dil modelleme, konuşma tanıma ve makine çevirisi gibi görevlerde önemli performans artışı sağlamlar ve karşılaştırmalı deneylerde başarılı sonuçlar elde etmiştir ([6], [7], [8]).

Şekil 3'te farklı sıralı modellerin (RNN, LSTM, GRU ve Transformers) yapısal farklılıklarını karşılaştırılmıştır.



Şekil 3. Farklı Mimarilerin Karşılaştırılması (RNN, LSTM, GRU ve Transformer) [10]

## Transformer Mimarisi ve Dönüm Noktaları

2017 yılında geliştirilen Transformer mimarisi, büyük dil modellerinin temel yapısını oluşturmuştur. Transformer, veri dizileri üzerinde paralel işlem yapabilen dikkat mekanizması (attention) ile çalışarak RNN'lerin sıralı işleme zorunluluğunu ortadan kaldırmıştır [9]. Bu yenilik, hem doğruluk hem de hız açısından önemli bir ilerleme sağlamıştır. Bu mimariyi temel alan Üretken Ön İşlemeli Dönüştürücü [Generative Pretrained Transformer (GPT)] ve Çift Yönlü Kodlayıcı Temsilleri [Bidirectional Encoder Representations from Transformers (BERT)] gibi modeller, doğal dil işleme alanında büyük başarılar elde etmiştir. Transformer tabanlı bu modeller, metin anlama, üretme ve özetleme gibi birçok görevde önceki mimarilere kıyasla daha iyi performans sergilemektedir.

### GPT ve BERT Modelleri

Transformer mimarisi, başlangıçta encoder-decoder yapısında tasarlanmış bir modeldir. Ancak zamanla, farklı doğal dil işleme görevlerine yönelik olarak bu mimarinin belirli bileşenleri kullanılarak özelleşmiş modeller geliştirilmiştir. BERT, yalnızca encoder katmanlarını kullanan bir modelken, GPT ise, sadece decoder katmanlarını temel alan bir modeldir. Bu temel fark, modellerin kullanım alanlarını belirlemiştir ve doğal dil işleme alanında farklı uygulamalara yön vermiştir [11].

#### GPT

GPT otoregresif bir model olup, metni üretirken her token'ı yani, kelime altı metin parçasını sırasıyla tahmin eder. GPT, tek yönlü self-attention (öz dikkat) mekanizması kullanarak yalnızca önceki token'ları dikkate alır ve gelecekteki token'ları tahmin eder. Transformer mimarisinin sunduğu paralel işlem yetenekleri sayesinde büyük veri kümeleri üzerinde ölçülebilir bir şekilde eğitilebilmektedir.

GPT serisinin gelişimi, dil modellerinin bağlam anlaması yeteneklerini güçlendirmeyi ve ek eğitim gerektirmeden farklı görevlerde etkili şekilde çalışmasını sağlamayı hedeflemiştir. Özellikle GPT-3 ile birlikte, few-shot learning (az örnekli öğrenme) adı verilen öğrenme yöntemi öne çıkmıştır. Few-shot learning, modelin yalnızca birkaç örnekle yeni görevleri öğrenmesine olanak tanır. Bu, dil modellerinin daha geniş kapsamlı ve esnek kullanım senaryolarına uyum sağlamasını mümkün kılan önemli bir adımdır [12]. Few-shot learning kavramı, istem mühendisliği başlıklarında detaylıca ele alınacaktır.

GPT, kendisine verilen istemin yapısına bağlı olarak yanıt üretir. Açık uçlu istemlerde daha uzun ve detaylı açıklamalar sunabildirken, belirli bir formatla yönlendirildiğinde daha kısa ve doğrudan yanıtlar verebilmektedir. Örneğin, bir soruya doğrudan yanıt vermesi beklenirken, bazen konuyu genişleterek bir blog yazısı formatında devam ettirebilmektedir. Bu eğilim, modelin otoregresif doğasından kaynaklansa da bağlamın tamamen kaybolduğu anlamına gelmemektedir. Model, verilen yönlendirmelere göre çıktısını optimize edebilir ve belirli formatlara uygun şekilde yanıt üretебilmektedir.

#### BERT

BERT, kelimelerin bağlamını hem önceki hem de sonraki kelimelerle birlikte değerlendirderek daha güçlü bir dil anlayışı geliştirmektedir [13]. GPT'den farklı olarak, metni üretmek yerine anlamlandır-

maya odaklanan bir model olup çift yönlü (bidirectional) dikkat mekanizması kullanılmaktadır.

BERT'in başarısının temelinde Maskelenmiş Dil Modeli [Masked Language Model (MLM)] ve Sonraki Cümle Tahmini [Next Sentence Prediction (NSP)] olmak üzere iki ana ön-eğitim (pre-training) stratejisi yer almaktadır.

### 1. Maskelenmiş Dil Modeli (MLM):

Bu yöntemde, modelin eğitim sırasında rastgele bazı token'lar maskelenir ve modelin eksik parçaları tahmin etmesi beklenir.

#### Örneğin:

"Ankara, [MASK] Cumhuriyeti'nin başkentidir."

Burada modelin "[MASK]" yerine "Türkiye" kelimesini getirmesi gereklidir. Bu sayede, BERT bağımlı hem önceki hem de sonraki kelimelerden öğrenerek daha güçlü bir anlam çıkarımı yapabilir.

### 2. Sonraki Cümle Tahmini (NSP):

Modelin, iki cümlenin birbirile bağlılığı olup olmadığını anlamasını sağlayan bir yöntemdir.

#### Örneğin:

**Cümle 1:** "Köpek bahçede koşuyordu."

**Cümle 2:** "Daha sonra eve girdi."

BERT, bu iki cümlenin gerçekten ardışık olup olmadığını anlamaya çalışır. Böylece NSP, özellikle metin ilişkilendirme, belge sıralama ve bilgi çıkarımı gibi görevlerde büyük avantaj sağlamlamaktadır.

## Büyük Modeller ve Küçük Modellerin Karşılaştırması

Büyük dil modelleri ve küçük modeller, kullanım senaryolarına göre farklı avantajlara sahiptir. Büyük modeller, karmaşık akıl yürütme ve uzun bağlam gerektiren işlemlerde üstün performans sergilerken, küçük modeller, daha düşük donanım gereksinimiyle daha hızlı yanıt üretir ve enerji verimliliği sağlar.

**Tablo 1. Büyük Modeller (LLM) ve Küçük Modellerin (SLM) Karşılaştırması**

Özellik	Büyük Modeller (LLM)	Küçük Modeller (SLM)
Parametre Sayısı	Milyarlarca ila trilyonlarca parametre içerir.	Milyonlarca ila birkaç milyar parametre içerir.
Bağlam Anlama	Çok adımlı mantık yürütmede üstündür.	Orta ölçekli bağlamlarda etkilidir.
Hesaplama Gücü	Yüksek donanım ihtiyacı vardır.	Daha düşük donanım gerektirir.
Yanıt Süresi	Daha uzun sürede işlem yapar.	Daha hızlı yanıt üretir.
Enerji Verimliliği	Daha fazla enerji tüketir.	Daha verimli çalışır.

Tablo 1'de büyük dil modelleri ile küçük modellerinin karşılaştırması gösterilmektedir. Büyük modeller daha fazla hesaplama gücü ve bellek kullanırken, küçük modeller düşük donanımlı sistemlerde daha verimli çalışmaktadır. Farklı model türlerinin güçlü ve zayıf yönlerini ortaya koyan Tablo 1, hangi modelin hangi durumda daha avantajlı olabileceğine dair fikir sağlamaktadır.

## Model Seçiminde Dikkat Edilmesi Gerekenler

Doğu büyük dil modeli seçiminde belirleyici kriterler, kullanım amacına ve mevcut kaynaklara bağlı olarak büyük ölçüde değişiklik gösterir. Bazı senaryolar büyük modelleri gerektirirken, bazıları küçük ve enerji verimli çözümleri tercih etmeyi gerektirebilir. Tablo 2, farklı kullanım alanları için önemli kriterleri ve en uygun örnek dil modellerini özetlemektedir.

**Tablo 2. Farklı Kullanım Alanları İçin Önemli Kriterleri Ve En Uygun Örnek Dil Modellerini Özetlemektedir.**

Kullanım Alanı	Önemli Kriterler	Örnek Model
<b>Genel Kullanım</b>	Geniş bilgi tabanı, çok yönlü dil yetenekleri	GPT-4, LLaMA 3 70B
<b>Uzun Metin İşleme</b>	Büyük bağlam penceresi, detaylı analiz	Claude 3, GPT-4o
<b>Mobil Uygulamalarda Kullanım</b>	Hafif yapı, düşük güç tüketimi, hızlı yanıt	Mistral 7B, Phi3
<b>Kodlama</b>	Programlama desteği, hata tespiti	DeepSeek Coder, StarCoder 15B
<b>Özel Görevler</b>	Spesifik kullanım için optimize edilmiş yapı	LLaMA 3 8B, Mistral 7B

## Parametre Boyutu ve Performans Karşılaştırması

Büyük dil modelleri, doğal dil işleme, kod üretimi, mantık yürütme ve çoklu modalite entegrasyonu gibi görevlerde etkili çözümler sunmaktadır. Ancak geleneksel anlayışa göre daha fazla parametreye sahip modellerin daha başarılı olacağı düşünülse de son araştırmalar bu varsayıımı sorgulamaktadır. Özellikle DeepSeek modelleri, yüksek parametre sayısına sahip olmadan da büyük modellere rakip olabilecek performans göstermesiyle dikkat çekmektedir.

## DeepSeek Modelinin Etkisi ve Parametre Verimliliği

Genel kabul, büyük dil modellerinin performansının doğrudan parametre sayısıyla ilişkili olduğu yönündedir. Ancak DeepSeek, düşük parametreli modellerin de güçlü sonuçlar elde edebileceği göstererek bu anlayışı kısmen değiştirmektedir. Yine de DeepSeek modelleri içinde bile daha fazla parametreye sahip versiyonlar genellikle daha iyi performans göstermektedir [14]. Dolayısıyla parametre sayısı ile başarı arasındaki genel pozitif korelasyon devam etmekle birlikte, verimli eğitim ve optimizasyon teknikleri sayesinde daha küçük modellerin de rekabetçi performans sunabileceği görülmektedir.

DeepSeek AI ekibi, geliştirdikleri DeepSeek-R1 modelini farklı boyutlarda distile ederek (**Qwen-1.5B**, **Qwen-7B**, **Qwen-14B**, **Qwen-32B gibi**), çok sayıda benchmark testi kapsamında karşılaştırmıştır [15]. Elde edilen sonuçlar, büyük modellerin her zaman üstün olmadığını göstermektedir. Küçültülen modellerin, daha az kaynak kullanarak büyük modellere yakın doğruluk oranları elde edebilmesi, distile modellerin pratik kullanım alanlarını genişletmektedir.

Tablo 3'te sunulan tablo, DeepSeek-R1 modelinin farklı boyutlarındaki distile edilmiş (hafifletilmiş) versiyonlarının mantık yürütme ve kodlama yetenekleri üzerindeki etkisini göstermektedir.

**AIME 2024, MATH-500, GPQA Diamond, LiveCode Bench ve CodeForces** gibi terimler, büyük dil modellerinin (LLM) farklı yeteneklerini değerlendirmek için kullanılan benchmark (karşılaştırma) testleridir. Her biri, modelin belirli bir görevdeki doğruluğunu ve performansını ölçmek için tasarlanmıştır.

**Pass@1**, modelin ilk denemede doğru yanıt üretme oranını gösterirken, **cons@64**, modele 64 farklı tahmin yapma şansı vererek en az bir tahminde doğru sonuca ulaşma oranını ölçen bir metriktir. Cons@64 metriği, modelin bir soruya karşı çoklu tahminden en doğrusunu içerebilme yeteneğini yansıtır ve genellikle pass@1'den daha yüksek bir doğruluk oranı sunar.

Şu bulgular dikkat çekmektedir [15]:

- **AIME 2024 (pass@1):** AIME (Artificial Intelligence Medical Evaluation), yapay zeka modellerinin tıbbi bilgi işleme ve mantıksal akıl yürütme yeteneklerini test eden bir benchmark'tır. Küçük modellerin (Qwen-1.5B: %28.9) başarısı sınırlıken, daha büyük modeller (Qwen-32B: %72.6, Llama-70B: %70.0) belirgin şekilde daha iyi performans sergilemiştir. Ancak, optimize edilmiş küçük modeller belirli senaryolarda rekabetçi kalabilmektedir.
- **AIME 2024 (cons@64):** AIME (Artificial Intelligence Medical Evaluation) testi kapsamında cons@64 metriği, modele aynı soru için 64 farklı tahminde bulunma şansı tanır ve bu tahminlerin ortalamasını değerlendirir. Bu yöntem, modelin tek seferde doğru yanıt verme olasılığını değil, çoklu tahminler arasında doğru cevabı içerebilme yeteneğini ölçer. Sonuçlar incelendiğinde, küçük modellerin (Qwen-1.5B: %52.7) cons@64 metriğiyle daha yüksek doğruluk oranlarına ulaşarak rekabetçi hale geldiği, büyük modellerin (Llama-70B: %86.7) ise doğruluklarını önemli ölçüde artırdığı görülmektedir. Ancak, bu yöntemin gerçek dünya kullanım senaryolarını tam olarak yansımadığı ve diğer metriklerle birlikte değerlendirilmesi gereği unutulmamalıdır.
- **MATH-500 (pass@1):** MATH-500, büyük dil modellerinin matematik problemlerini çözme yeteneğini test eden bir benchmark'tır. Büyük modellerin (Qwen-14B: %92.8, Qwen-32B: %94.5) karmaşık işlemleri daha iyi çözdüğü gözlemlenmiştir.
- **GPQA Diamond (pass@1):** GPQA (General-Purpose Question Answering), modelin genel bilgi ve mantıksal akıl yürütme yeteneğini ölçer. Diamond seviyesi, en zor sorular içeren veri setini temsil eder. Baseline modeller olan GPT-4 (%85.2) ve Claude 3 (%83.9) en yüksek performansı gösterirken, DeepSeek-7B gibi optimize edilmiş modeller (%75.3) daha düşük parametreyle rekabetçi bir başarı yakalamayı başarmıştır.

- LiveCode Bench (pass@1):** LiveCode Bench, modelin yazılım kodu üretme ve kod tamamlama becerisini test eden bir benchmark'tır. Büyük modeller (Qwen-32B: %57.2, Llama-70B: %57.5) daha başarılı olmasına rağmen, düşük parametreli DeepSeek modelleri de verimli kod üretimi sağlayarak güçlü sonuçlar elde etmiştir.
- CodeForces (rating):** CodeForces, rekabetçi programlama platformlarından biridir. Rating metriği, modelin algoritmik problem çözme ve rekabetçi programlama yarışmalarındaki başarısını ölçer. OpenAI-01-mini 1820 puan ile en iyi sonucu elde ederken, Qwen-32B ve Llama-70B rekabetçi seviyede kalmıştır. Optimize edilmiş DeepSeek-7B ise belirli görevlerde büyük modellere yakın sonuçlar vermiştir.

**Tablo 3. DeepSeek-R1'in Farklı Boyutlardaki Distile Edilmiş Versiyonlarının Mantık Yürütme ve Kodlama Yetenekleri Üzerindeki Karşılaştırması [15]**

Model	Benchmark Testleri ve Metrikleri					
	AIME 2024w		Math-500	GPQA Diamond	LiveCode Bench	CodeForces
	pass@1	cons@64	pass@1	pass@1	pass@1	rating
GPT-4o-0513	9.3	13.4	74.6	49.9	32.9	759
Claude-3.5-Sonnet-1022	16.0	26.7	78.3	65.0	38.9	717
OpenAI-o1-mini	63.6	80.0	90.0	60.0	53.8	1820
QwQ-32B-Preview	50.0	60.0	90.6	54.5	41.9	1316
DeepSeek-R1-Distill-Qwen-1.5B	28.9	52.7	83.9	33.8	16.9	954
DeepSeek-R1-Distill-Qwen-7B	55.5	83.3	92.8	49.1	37.6	1189
DeepSeek-R1-Distill-Qwen-14B	69.7	80.0	93.9	59.1	53.1	1481
DeepSeek-R1-Distill-Qwen-32B	72.6	83.3	94.3	62.1	57.2	1691
DeepSeek-R1-Distill-Llama-8B	50.4	80.0	89.1	49.0	39.6	1205
DeepSeek-R1-Distill-Llama-70B	70.0	86.7	94.5	65.2	57.5	1633

Bu veriler, distile modellerin büyük modellere kıyasla daha az hesaplama gücü gerektirerek verimli çözümler sunduğunu göstermektedir [15]. Özellikle distile edilmiş modeller, belirli görevlerde büyük modellere yakın performans sergileyerek pratik kullanım için güçlü alternatifler oluşturabilir. Bu durum, kaynak tüketimi düşük yapay zeka çözümleri için zemin hazırlamaktadır.

## GPT Modellerinden ChatGPT'ye Geçiş ve Talimat Ayarlama'nın (Instruction Tuning) Rolü

GPT modellerinin gelişimi, başlangıçta büyük ölçüde otoregresif dil üretimi üzerine odaklanmıştır. GPT-2 ile birlikte, dil modeli çeşitli görevleri yerine getirebilse de çıktılarının kontrol edilebilirliği ve doğruluğu açısından bazı sınırlamalar barındırmaktaydı. Bu süreçte, belirli görevlerin yerine getirilmesi için özenle tasarlanmış istemlere duyulan ihtiyaç, istem mühendisliği (prompt mühendisliği) kavramının doğmasına neden olmuştur.

İstem mühendisliğinin önemi, özellikle GPT-3 ve ChatGPT ile daha belirgin hale gelmiştir. ChatGPT, Talimat Ayarlama (Instruction Tuning) adı verilen yöntemle eğitilerek, kullanıcının verdiği talimatları daha iyi anlamaya ve uygulamaya başlamıştır [16]. Talimat Ayarlama, modele belirli talimatlara nasıl yanıt verilmesi gerektiğini öğreten bir süreçtir. Bu sayede model, yalnızca sonraki kelimeleri tahmin eden bir yapı olmaktan çıkıp, daha bilinçli ve yönlendirilmiş çıktılar üretebilen bir sisteme evrilmiştir.

Talimat Ayarlama'nın en büyük avantajlarından biri, modelin bağlam anlayışını ve görev uyumunu artırmasıdır. GPT-2 döneminde istem mühendisliği deneyimsel seviyedeyken, bu yöntemin verimli bir şekilde uygulanabilmesi için modelin doğrudan eğitilmesi gerektiği anlaşılmıştır. Talimat Ayarlama ile birlikte, model verilen komutlara daha iyi yanıt vererek doğal dil işleme görevlerinde gelişme göstermiştir [17].



# Büyük Dil Modelerinin Kullanım Alanları

Büyük dil modelleri, doğru şekilde yönlendirildiğinde günlük hayattan profesyonel iş süreçlerine kadar pek çok alanda etkili çözümler sunabilir. Aşağıda, yaygın kullanım alanları özetlenmiştir:

- Kod Düzenleme ve Tamamlama:** Hatalı veya eksik kodları düzelterek daha verimli hale getirmek, açıklamalar ekleyerek anlaşılmasını kolaylaştırmak mümkündür. Örneğin, Python'da bir fonksiyonun performansını iyileştirmek için öneriler almak.
- Belge Öztleme ve İçerik Üretimi:** Uzun raporları kısa ve öz bilgilere dönüştürmek, teknik dokümantasyon yazmak veya akademik makale taslakları oluşturmak için kullanılabilir. Örneğin, 50 sayfalık bir araştırma makalesini 500 kelimelik bir özet haline getirmek.
- E-posta ve Rapor Yazımı:** Resmî yazışmalar, müşteri geri bildirimleri ve iş başvuruları gibi konularda etkili içerikler oluşturmak mümkündür. Örneğin, bir iş başvurusu için etkili bir kapak mektubu hazırlamak.
- Müşteri Hizmetleri Otomasyonu:** Sıkça sorulan sorulara hızlı yanıtlar verebilen otomatik destek sistemleri oluşturulabilir. Örneğin, bir şirketin chatbot'unun müşteri taleplerine anında ve doğru şekilde yanıt vermesi.
- Sosyal Medya İçerik Üretimi:** Blog yazıları, ürün tanıtım metinleri, başlık önerileri ve sosyal medya gönderileri hazırlamak için kullanılabilir. Örneğin, Instagram için dikkat çekici bir ürün açıklaması yazmak.
- Veri Analizi ve Görselleştirme:** Büyük veri kümelerinden anlamlı içgörüler elde etmek ve grafik veya tablo önerileri sunmak mümkündür. Örneğin, bir Excel dosyasındaki müşteri verilerini analiz ederek en iyi satış stratejisini belirlemek.
- Proje Danışmanlığı:** Araştırma ve geliştirme süreçlerine rehberlik etmek, proje yönetim planları oluşturmak gibi konularda yardımcı olabilir. Örneğin, yapay zeka tabanlı bir müşteri hizmetleri chatbot'u geliştirme süreci için proje planı oluşturmak.
- İş Planı ve Stratejik Planlama:** Yeni projeler için iş planı oluşturmak, stratejik hedefler belirlemek ve analizler yapmak mümkündür. Örneğin, yapay zeka tabanlı bir araştırma projesi için beş yıllık iş planı hazırlamak ve SWOT analizi yapmak.

Bu kullanım alanları, büyük dil modellerinin sunduğu geniş olanaklardan sadece birkaçıını göstermektedir. Doğru yönlendirildiğinde, bu modeller kod yazımından stratejik planlamaya, içerik üretiminden veri analizine kadar birçok süreçte verimliliği artırabilir ve karar alma süreçlerini hızlandıracaktır. Ancak, bu sistemlerin yaygın kullanımına rağmen, etik, güvenlik ve teknik açıdan bazı önemli zorlukları da beraberinde getirdiği unutulmamalıdır. Bu zorluklar, büyük dil modellerinin gelişimi açısından dikkate alınması gereken konular arasında yer almaktadır.

# Büyük Dil Modellerinin Güçlü ve Zayıf Yönleri

Büyük dil modelleri, geniş bilgi kapsamı ve gelişmiş analiz yetenekleri sayesinde birçok alanda etkili çözümler sunarken, bazı teknik ve etik zorluklarla da karşılaşmaktadır. Tablo 4'te, bu modellerin güçlü ve zayıf yönleri özetlenmiş, olası çözüm önerileri sunulmuştur.

**Tablo 4. Büyük Dil Modellerinin Güçlü ve Zayıf Yönleri**

Özellik	Güçlü Yönler	Zayıf Yönler	Çözüm Önerisi
<b>Bağlam Anlama</b>	Model, çok dilli destek ve geniş veri setleri sayesinde yüksek doğruluk sunar.	Düşük kaynaklı dillerde bağımlı yanlış anlayabilir ve anlam kayıpları yaşanabilir.	Özelleştirilmiş veri setleri kullanılarak modelin kültürel bağımlı daha iyi kavraması sağlanmalıdır.
<b>Karmaşık Problem Çözme</b>	Model, mantıksal akıl yürütme ve çok adımlı analizlerde başarılıdır.	Bazen yanlış akıl yürütme yaparak sonuçları genelleyebilir ve hatalı çıkarımlar yapabilir.	Düşünce zinciri yöntemi ve bağlama göre yönlendirilmiş istemler kullanılarak modelin doğruluğu artırılmalıdır.
<b>Multi-Modal İşleme</b>	Model, metin, görsel ve ses gibi farklı veri türlerini işleyebilir ve birleştirilebilir.	Bazı görevlerde belirli veri türlerine karşı daha az duyarlıdır ve doğruluk oranı düşebilir.	Model mimarisi, farklı veri türlerine özel olarak optimize edilmeli ve çoklu modaliteyi daha iyi işleyebilmesi için geliştirilmelidir.
<b>Ölçeklenebilirlik</b>	Model, büyük veri işleme kapasitesine sahiptir ve optimizasyon teknikleriyle daha hızlı çalışabilir.	Yüksek hesaplama gücü gerektirdiği için maliyet açısından verimli olmayabilir.	Modellerin sıkıştırılması (distilasyon) veya daha küçük dil modellerinin kullanılmasıyla hesaplama maliyetleri azaltılabilir.
<b>Yanlılık ve Adil Çıktılar</b>	Büyük veri setlerinden faydalananak geniş perspektifler sunabilir.	Eğitim verilerindeki önyargıları öğrenerek taraflı içerik üretebilir ve adil olmayan sonuçlar ortaya çıkarabilir.	Daha dengeli ve çeşitli veri setleriyle model eğitilmeli ve yanlışlık tespit mekanizmaları uygulanmalıdır.
<b>Güvenlik ve Gizlilik</b>	Model, veri analizi ve otomatik karar alma süreçlerinde önemli avantajlar sunabilir.	Kişisel verileri yanlışlıkla öğrenebilir ve manipülasyona açık olabilir.	Anonimleştirme yöntemleri, güvenlik filtreleri ve veri koruma politikaları uygunlarak kullanıcı gizliliği korunmalıdır.

Bu modellerin daha güvenilir, ölçeklenebilir ve etik kullanımı için yanlışlık tespit mekanizmaları geliştirilmeli, güvenlik önlemleri artırılmalı ve daha hafif, maliyet etkin çözümler üretilmelidir.

# İstem Mühendisliğinin Tanımı ve Önemi

ChatGPT ve benzeri modellerin gelişimiyle birlikte, büyük dil modellerinin potansiyelinden tam anlamıyla yararlanabilmek için istemlerin (prompt'ların) doğru şekilde tasarılanması gerektiği anlaşılmıştır. Bu süreçte, öncelikle problemin doğru tanımlanması gereklidir. Modelden beklenen çıktıının formatı (metin, liste, tablo vb.) ve kullanım bağlamı açık bir şekilde belirlenmelidir. Problemi doğru tanımlamak, modelin çözüm üretme kapasitesini doğrudan etkiler. Bunun yanı sıra, modelin hangi senaryoda ve hangi amaca hizmet edeceğini netleştirmelidir. Modelin çıktılarının doğruluğunu, bağlama uyumunu ve görev başarısını artırmak için istem mühendisliği kritik bir rol oynamaktadır. Bu sebeple, istem mühendisliği, model çıktılarının optimize edilmesi ve etkili çözümler üretilmesi için sistematik bir yaklaşım sunmaktadır.

## Doğru İstem Yapısını Belirlemek

Bir problemin tanımlanmasının ardından, modele sunulan istemlerin yapısı kritik bir rol oynar. Farklı stratejiler farklı durumlar için etkili olabilir:

- **Sıfır Atış İstemleme (Zero-shot Prompting):** Modele hiçbir örnek sunmadan yanıt ürettilir. Bu yöntem, modelin eğitim esnasında kazandığı genel bilgi birikiminden faydalananarak sonuç üretmesine olanak tanır [12].
- **Tek Atış ve Az Örnekli İstemleme (One-shot & Few-shot Prompting):** Bir veya birkaç örnek sunularak modelin bağlama uygun çözüm sunması sağlanır. Bu teknik, modelin o anki girdiyle daha uyumlu çıktılar vermesine yardımcı olur [19]. Ayrıca model çıktısının belirli bir formatı takip etmesi istenen durumlarda ilgili formatı örnekleme açısından faydalıdır.
- **Düşünce Zinciri İstemleri (Chain-of-Thought Prompts):** Modelin adım adım mantıksal akış oluşturarak karmaşık sorunları çözmeyi sağlar. Bu yöntem, modelin akıl yürütme kapasitesini belirgin bir şekilde güçlendirilebilir [20].

Şekil 4'te görüldüğü gibi, Düşünce Zinciri (CoT) yöntemi, sıfır-atış ve az örnekli yaklaşımlara ayrılarak sınıflandırılmaktadır.



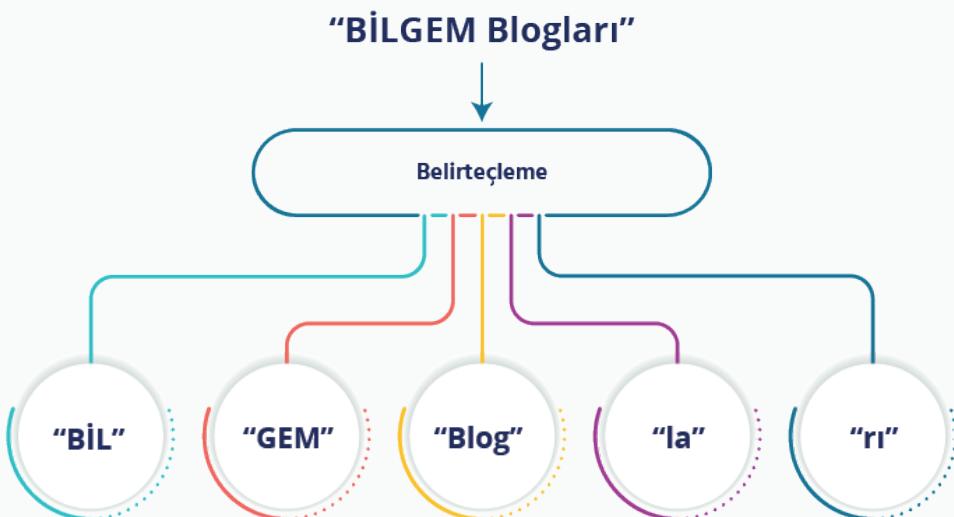
Şekil 4. Düşünce Zinciri (CoT) Yöntemlerinin Sınıflandırılması [18]

Bu tekniklerin detaylı analizi, yazının ilerleyen bölümlerinde **Gelimiş İstem Teknikleri** başlığı altında ele alınacaktır.

## İstem Kodlama ve Sıkıştırma Teknikleri

İstemin model tarafından etkili bir şekilde işlenmesi için doğru kodlama (encoding) kritik öneme sahiptir.

- **Belirteçleme (Tokenizasyon):** Girdi metni, modelin işleyebileceği daha küçük parçalara ayrılarak kodlanır. Bu yöntem, metindeki anlamsal ilişkilerin daha iyi kavranmasını sağlayarak modelin doğruluk oranını artırır ve çıktılarının daha tutarlı olmasına yardımcı olur. Şekil 5'te, bir metnin belirteçleme sürecinden nasıl geçtiği ve kelime bileşenlerine ayrılma örneği gösterilmektedir. Tokenizasyon, özellikle büyük dil modellerinde bağlamsal anlam bütünlüğünü korumak ve daha etkili dil işleme sağlamak için kritik bir adımdır.



Şekil 5. Tokenizasyon (Belirteçleme) Süreci ve Örnek Kelime Bölünmesi

- **Embedding Yapıları:** Girdi metni, model tarafından işlenebilir anlamlı vektörlere dönüştürülür.

## Etkili İstem Yazım Teknikleri

Nasıl etkili bir istem yazılıcasının bilinmesi, büyük dil modellerinin potansiyelinden tam anlamıyla yararlanmanın anahtarlarından biridir. İyi bir istem, açık talimatlar vererek modelin karmaşık sorulara doğru çözümler sunmasını sağlar. Bu bölümde, etkili istem yazımına yönelik temel kurallar, örnekler ve stratejiler ele alınmaktadır.

### Etkili İstem Yazımında Temel Kurallar

#### Kısa ve Net Olunmalıdır

Modelin net ifadeleri daha iyi anladığını unutmayın. Belirsiz veya çok genel ifadeler kullanıldığında modelin doğru yanıt verme ihtimali azalır [21].

### Örnek:

- **Yanlış:** "Bana güzel bir yazı yaz."
- **Doğru:** "Türk mutfağının tarihçesini anlatan 300 kelimelik bir yazı yaz."

Açık ve kesin ifadeler kullanarak modelin yanıtını dar bir çerçevede oluşturması sağlanabilir [22].

### Bağlam ve Amaç Eklenmelidir

Modelden belirli bir konuda veya tarzda yanıt almak isteniliyorsa, bunu açıkça belirtmek gerekmektedir [21].

### Örnek:

- **Yanlış:** "Yapay zeka hakkında bir şeyler yaz."
- **Doğru:** "Bir gazeteci gibi yaz ve yapay zekanın sağlık alanındaki kullanımını ele al."

Bağlam eklemek, özellikle spesifik alanlardaki görevlerde modelin performansını artırmaktadır [16].

### Modelin Yanımı Sınırlandırılmalıdır

Bazen modelin çok uzun veya alakasız yanıtlar üretmesini önlemek gereklidir. Bunu yapmanın en iyi yolu, cevap formatını veya uzunluğunu sınırlandırmaktır.

### Örnek:

- **Yanlış:** "Bana bir bilgisayar oyunu yaz."
- **Doğru:** "Bana kendi başına çalıştırabileceğim bir betik halinde Python dilinde basit bir yılan oyunu yaz."

### Geri Bildirim Döngüsü Kullanılmalıdır

Modelin verdiği yanıtları değerlendirerek istem geliştirmeniz mümkündür. Model yanıtlarına geri bildirim vererek beslediğiniz bir yaklaşım, modelin yanıtlarını daha doğru ve etkili hale getirir [23].

### Örnek:

- **İlk istem:** "Yapay zekanın iş dünyasındaki etkisini açıkla."
  - **Modelin Yanımı:** "Yapay zeka, iş dünyasında otomasyonu artırarak verimliliği artır."

**Kullanıcı Geri Bildirimi:** "Yanıt çok genel oldu. Daha spesifik örnekler vererek açıklar mısın?"

- **Geliştirilmiş istem:** "Yapay zekanın iş dünyasındaki etkisini, müşteri hizmetleri, finans ve üretim sektörlerinden spesifik örnekler vererek açıkla."
  - **Modelin Yeni Yanımı:** "Müşteri hizmetlerinde, yapay zeka destekli chatbot'lar müşteri taleplerine hızlı yanıt vererek şirketlerin maliyetlerini düşürüyor. Finans sektöründe, algoritmik ticaret sistemleri büyük veri setlerini analiz ederek yatırım kararlarını optimize ediyor..."

# Gelişmiş İstem Teknikleri

Gelişmiş istem teknikleri, büyük dil modellerinin performansını artırmak ve karmaşık problemlerin çözümünde daha etkili hale gelmesini sağlamak için geliştirilmiştir. Bu teknikler, modelin mantıksal akıl yürütme becerilerini güçlendirmek, yaratıcı çözümler üretmesini sağlamak ve belirli görevlerde daha yüksek doğruluk elde etmesine yardımcı olmak amacıyla tasarlanmıştır.

## Düşünce Zinciri [Chain-of-Thought (CoT)]:

Düşünce Zinciri [Chain-of-Thought (CoT)], modelin karmaşık problemleri adım adım çözmeyi sağlayan bir yöntemdir. **Bu yaklaşım, modelin yanıtlarını aşamalandırarak ara adımlar üretmesini zorunlu kılar ve böylece doğruluğun artmasını sağlar.** Model, yalnızca nihai cevabı sunmak yerine, belirli bir mantıksal akış içerisinde ilerleyerek daha güvenilir ve tutarlı yanıtlar üretебilir.

Özellikle **matematik problemleri, mantık tabanlı sorular ve çok aşamalı analiz gerektiren görevlerde**, CoT yöntemi modelin daha doğru sonuçlar üretmesini sağlar. Şekil 6'da, standart yönendirme ile CoT yöntemi arasındaki farklar açıkça görülmektedir. Standart yöntemde model, doğrudan nihai cevabı tahmin etmeye çalışırken, CoT yöntemi ara adımları hesaplayarak doğruluğu artırmaktadır.



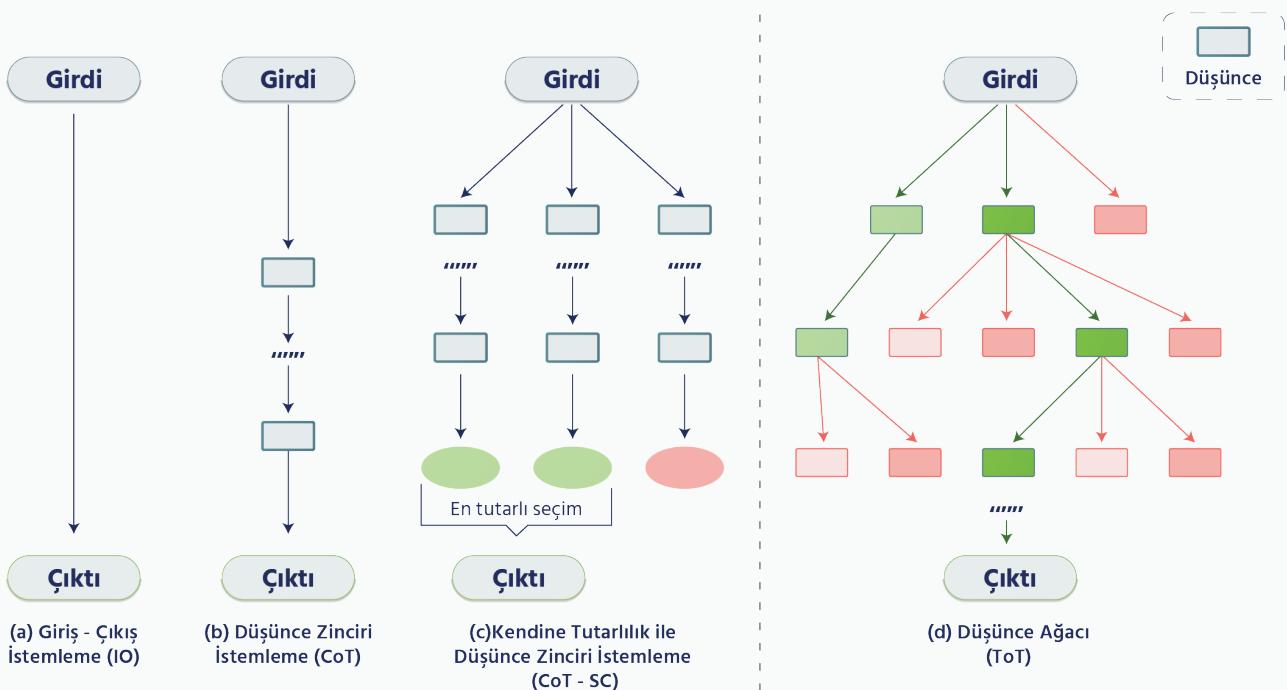
Şekil 6. Standart İstemleme ile Düşünce Zinciri İstemleme Karşılaştırması [20]

Araştırmalar, CoT yönteminin dil modellerinin problem çözme yeteneklerini önemli ölçüde geliştirdiğini ve hata oranlarını düşürdüğünü göstermektedir [20]. Bu yöntem, modelin sadece bir sonuca ulaşmasını değil, aynı zamanda bu sonuca **hangi mantıksal süreçleri izleyerek** ulaştığını da açıklamasını sağlayarak **akıl yürütme kapasitesini** güçlendirmektedir.

## Düşünce Ağacı [Tree-of-Thought (ToT)]:

Düşünce Ağacı [Tree-of-Thought (ToT)] yöntemi, modelin bir problemi adım adım çözerek farklı seçenekleri değerlendirmesine olanak tanır. Bu yaklaşım, modelin yalnızca tek bir çözüm yolu izlemesini değil, alternatif yolları keşfetmesini de sağlar. Düşünce düğümleri ve bunlar arasındaki bağlantılar sayesinde model, mantıksal bir akış içinde ilerleyerek karmaşık problemleri daha yüksek doğrulukla ve daha düşük maliyetle çözebilir [24].

Şekil 7'de, Düşünce Ağacı yöntemi ile klasik Giriş/Çıkış (I/O), Düşünce Zinciri ve Kendine Tutarlılık ile Düşünce Zinciri (CoT-SC) yöntemleri karşılaştırılmaktadır. Düşünce Ağacı yöntemi, alternatif düşünce yollarını keşfetme ve en uygun çözümü belirleme esnekliği sunarak problem çözme süreçlerinde önemli bir avantaj sağlamaktadır.



*Şekil 7. Düşünce Ağacı Yöntemi ile Farklı İstemleme Stratejilerinin Karşılaştırılması [24]*

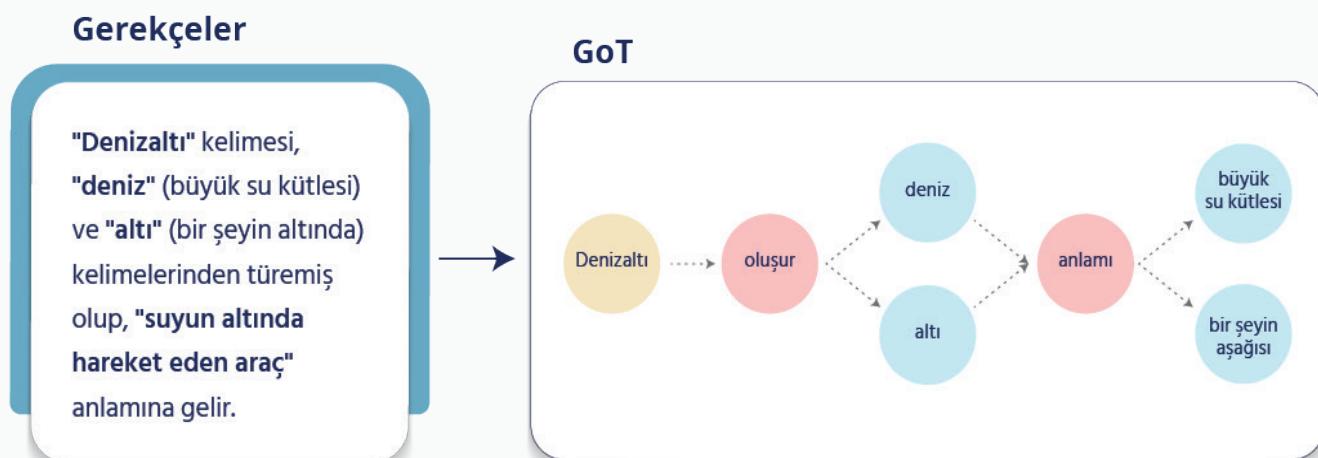
Bu çok yönlü yapı, modelin yalnızca tek bir çözüm yolunu takip etmek yerine, farklı alternatifleri değerlendирerek en uygun çözümü üretmesini sağlar. Böylece, daha yüksek doğruluk oranı ve problem çözme süreçlerinde esneklik avantajı elde edilir.

## Düşünce Grafi [Graph-of-Thought (GoT)]:

Düşünce Grafi [Graph-of-Thought (GoT)] yöntemi, bilgileri bir ağaç yapısından daha karmaşık bir graf yapısına dönüştürerek organize etmeye dayanır. Bu yöntemde, düşünce birimleri (düğümler) ve bunlar arasındaki ilişkiler (kenarlar) oluşturularak modelin daha kapsamlı bir akıl yürütme süreci gerçekleştirmesi sağlanır.

Şekil 8'de, Düşünce Grafi yöntemi kullanılarak kelime türetme sürecinin nasıl yapılandırıldığı gösterilmektedir. GoT yöntemi, karmaşık problemlerde yüksek doğruluk ve düşük işlem maliyeti

sunan bir yapıdır. Düşünce Ağacı yöntemi belirli mantıksal adımlarla ilerlerken, GoT çok boyutlu ve bağıntılı düşünme modelini benimserek farklı çözüm yollarını bir arada değerlendirdir.



Şekil 8. Düşünce Grafiği (GoT) ile Kelime Türetme Sürecinin Görselleştirilmesi [25]

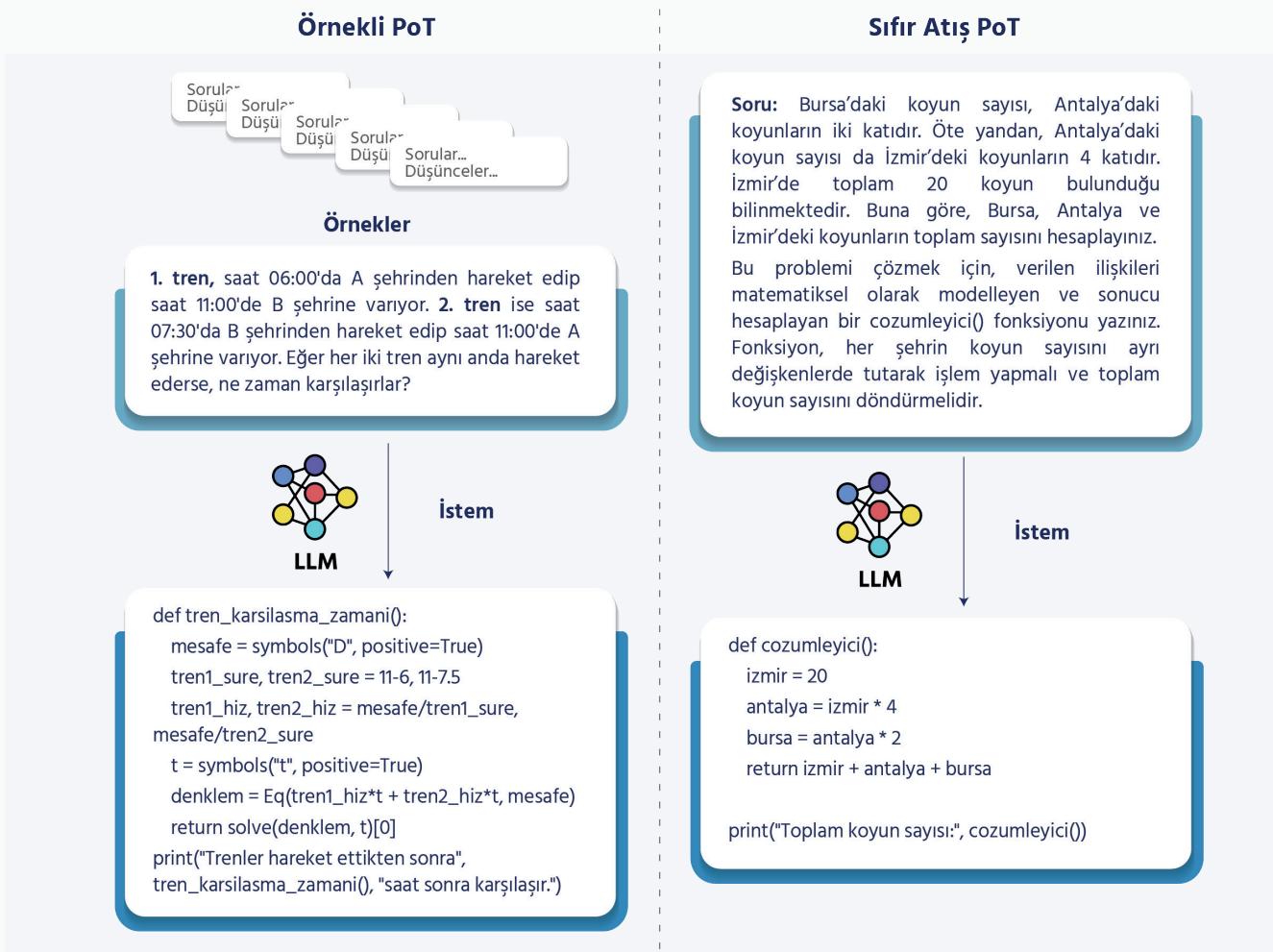
Araştırmalara göre, GoT yöntemi, ToT'ye oranla %62 daha yüksek doğruluk sağlarken işlem maliyetlerini %31 oranında azaltmaktadır [23]. Şekil 9'da da görüldüğü gibi, GoT yöntemi klasik ağaç yapılarına kıyasla daha esnek ve çok yönlü bir problem çözme süreci sunmaktadır.

### Düşünce Programlama [Program-of-Thought (PoT)]:

Düşünce Programlama [Program- of-Thought (PoT)], dil modellerinin **programlama tabanlı mantıksal akışlar oluşturarak** çıktı üretmesini sağlayan bir yöntemdir. Bu yaklaşım, **hesaplama ve akıl yürütme süreçlerini ayırtırarak**, modelin çözüm sürecini **kod benzeri bir yapıya** dönüştürmesine olanak tanır.

Şekil 9'da, PoT'nin problem çözme sürecine nasıl entegre edildiği gösterilmektedir. Sol tarafta, model geçmiş örneklerden öğrenerek **hız, mesafe ve zaman arasındaki ilişkiyi adım adım kod olarak ifade etmektedir**. Sağ tarafta ise verilen bir problem analiz edilerek Python fonksiyonu içinde değişkenler tanımlanmakta ve çözüm süreci kod aracılığıyla yürütülmektedir. Bu yöntem, yalnızca bir sonuca ulaşmayı değil, **çözüm sürecini matematiksel işlemler ve programlama adımlarıyla şeffaf hale getirmeyi amaçlamaktadır**. Çalışmalar, PoT'nin geleneksel yaklaşımına kıyasla mantıksal problem çözme ve hesaplama süreçlerinde daha yüksek doğruluk sağladığını göstermektedir [26].

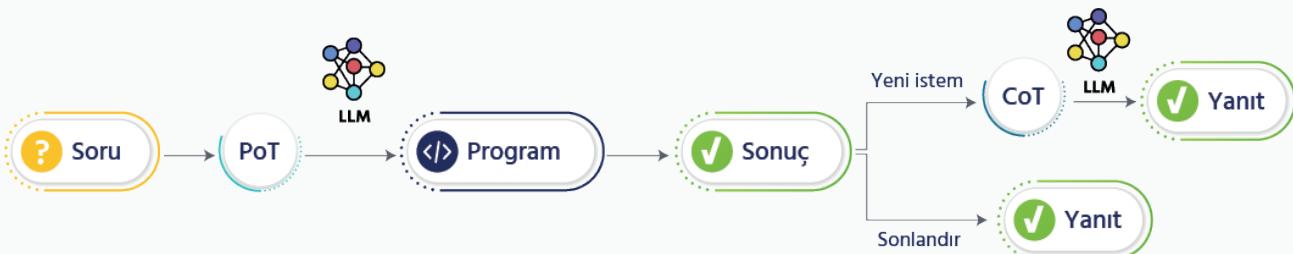
<Large\_Language\_Model/>



Şekil 9. Düşünce Programlama (PoT) Yöntemi ile Programatik Problem Çözme Sürecinin Görselleştirilmesi [26]

Deneysel, PoT'nin Düşünce Zinciri yöntemine kıyasla yaklaşık %12 daha doğru sonuçlar ürettiğini ve matematiksel problem çözme görevlerinde en güncel performansa (SoTA) ulaştığını ortaya koymaktadır [26].

Şekil 10'da, PoT ve CoT'nın çok aşamalı akıl yürütme sürecinde bir arada kullanıldığı görülmektedir. Bu modelleme sürecinde, PoT önce programatik bir çözüm üretir, ardından elde edilen sonuçlar ek istemleme ile CoT sürecine aktarılır. Böylece model, karar mekanizmasını genişleterek daha yapılandırılmış ve güvenilir yanıtlar üretir.



Şekil 10. PoT ve CoT Yöntemlerinin Çok Aşamalı Akıl Yürütme İçin Birleşimi [26]

Özellikle finansal hesaplama, sayısal analiz ve mantık tabanlı problem çözme gibi alanlarda PoT'nin CoT ile birlikte kullanıldığına en yüksek verimi sağladığı görülmektedir. Şekil 10'da gösterildiği gibi, bu yöntem, karmaşık problemlerin çözümünü daha sistematik ve tutarlı hale getirmektedir.

## Gelişmiş Tekniklerin Avantajları ve Kullanım Alanları

Bu gelişmiş teknikler, büyük dil modellerinin daha esnek, yaratıcı ve mantıksal olarak tutarlı bir şekilde çalışmasını sağlamak açısından kritik öneme sahiptir. Gelecekte, bu yaklaşımın daha da optimize edilmesiyle, dil modellerinin karmaşık görevleri başarıyla yerine getirmesi beklenmektedir.

Tablo 5'te, farklı problem türleri için kullanılan gelişmiş istemleme teknikleri ve bu tekniklerin hangi alanlarda uygulandığı özetlenmiştir.

**Tablo 5. Gelişmiş İstemleme Teknikleri ve Kullanım Alanları**

Problem Türü	Kullanılan Yöntem	Kullanım Alanı
<b>Çok Adımlı Akıl Yürütme</b>	Düşünce Zinciri	Matematik, mantık tabanlı soru-cevaplama, karar destek sistemleri
<b>Matematiksel Problem Çözme</b>	Düşünce Programlama	Sayısal analiz, bilimsel hesaplamalar
<b>Kod Üretimi</b>	Düşünce Programlama	Yapay zeka destekli kod üretimi, hata ayıklama, otomasyon sistemleri
<b>Büyük Veri Analizi ve Karar Ağaçları</b>	Düşünce Ağacı	Büyük ölçekli veri setleri, optimizasyon problemleri, finansal modelleme
<b>Doğal Dil Anlama ve Diyalog Yönetimi</b>	Düşünce Grafları	Akıllı sohbet botları, doğal dil işleme tabanlı öneri sistemleri
<b>Finansal Hesaplama ve Raporlama</b>	Düşünce Zinciri + Düşünce Programlama	Finansal raporlama, bütçe analizi, ekonomik tahminleme
<b>Görsel Anlamlandırma ve Veri Entegrasyonu</b>	Düşünce Grafiği + Çok Modlu Dil Modeli	Görsel veri analizi, bilgisayarla görüş, robotik algılama

## Örneklerle İstemleme Yöntemleri

**İstem mühendisliği, büyük dil modellerinden en verimli ve doğru çıktıları elde etmek için kullanılan tekniklerin bütünüdür.** Aşağıda, etkili istem yazımı için temel yöntemler sıralanmış olup, her biri için verilen örnekler, geliştirilmeye devam edilen BİLGEM yapay zeka asistanı kullanılarak sunulmuştur.

TÜBİTAK BİLGEM Yazılım Teknolojileri Araştırma Enstitüsü tarafından geliştirilen bu araç, kurum içi süreçlerde ve uzmanlık alanlarında asistanlık sağlama amacıyla kapalı ağıda çalışan bir yapay zeka chat uygulamasıdır. Şu an için, Gemma 27B büyük dil modeli temel alınarak özel kullanım senaryo-

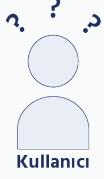
lara göre geliştirilmektedir. Görstellerde sunulan istemler yalnızca dil modellerinin genel kullanım senaryolarını göstermek için hazırlanmış olup, gerçek kullanımda farklı bağlamlara uyarlanabilir ve farklı çıktılar verebileceği göz önünde bulundurulmalıdır.

## 1. Belirginlik Temelli İstemleme (Specificity-Based Prompting)

Modelin daha doğru ve tutarlı yanıtlar verebilmesi için belirsiz ve genel ifadeler yerine açık, **yönlendirici** ve **detaylı istemler** kullanmayı amaçlar. Modelin üreteceği çıktıının net bir çerçeveye sahip olması, yanıtın **daha anlamlı ve bağlama uygun** olmasını sağlar.

Şekil 11'de de görüleceği gibi, model **belirsiz bir komutla yönlendirildiğinde**, ortaya çıkan **yanıt genellikle yüzeysel ve genel olabilir**.

Model, genel bir yapay zeka tanımı mı verecek yoksa doğal dil işleme alanına odaklanarak spesifik bir açıklama mı yapacak?

 Yeni Sohbet


 Yanlış Örnek

Bana yapay zeka hakkında bilgi ver

**BİLGEM Yapay Zeka Asistanı**  
Yapay zeka(YZ), makinelerin insan zekasını taklit etme yeteneğidir. Bu, öğrenme, problem çözme, karar verme ve dil anlaması gibi görevleri gerçekleştirmelerini içerir.  
YZ, şu alt alanlara ayrılır:

- Dar yapay zeka (Narrow AI):** Belirli bir görevi yerine getirmek için tasarlanmıştır. Örneğin, spam filtreleme, görüntü tanıma veya otomatik çeviri.

 Doğru Örnek

Yapay zekanın doğal dil işleme alanındaki etkisini, 200 kelimeyle açıkla.

**BİLGEM Yapay Zeka Asistanı**  
Yapay zeka(YZ), doğal dil işleme (NLP) alanında devrim yarattı. YZ tabanlı NLP modelleri, metinleri anlama, yorumlama ve üretme konusunda insan benzeri performans sergiliyor. Bu modeller, makine çevirisisi, metin özetlemesi, soru cevaplama ve duygusal analizi gibi görevlerde kullanılıyor. Ayrıca, sohbet botları, sanal asistanlar ve akıllı arama motorları gibi uygulamalarda da yer alıyor.

Şekil 11. Belirginlik Temelli İstemleme

Belirgin temelli istemler, kullanıcının öğrenmek istediği bilgiyi **tam olarak karşılamayabilir**. Öte yandan, **daha spesifik ve yönlendirici bir istem kullanıldığında**, model belirli bir konuya (örneğin,

Doğal Dil İşleme - NLP) ve belirlenmiş bir uzunlukta (örneğin, 200 kelime) yanıt üretmeye yönlendirilmiştir. Bu sayede, model **daha hedefe yönelik, detaylı ve bağlama uygun** bir çıktı sunabilmiştir.

**Belirginlik Temelli İstemler**, modelin **daha yapılandırılmış, net ve amaca uygun yanıtlar üretmesini** sağlayan önemli bir tekniktir. Bu yöntem, belirsiz ve genel ifadelerden kaçınarak modelin verilen görevi daha iyi anlamasına ve yönlendirilmesine yardımcı olur.

## 2. Talimat Temelli İstemleme (Instruction-Based Prompting)

Modelin belirli bir görevi nasıl yerine getirmesi gerektiğini net bir şekilde tanımlamayı amaçlar. Açık ve yönlendirici talimatlar, modelin beklenen çıktıyı doğru formatta üretmesine yardımcı olur. Belirsiz veya eksik komutlar ise modelin yanlış, yetersiz ya da bağlamdan kopuk yanıtlar vermesine neden olabilir.

Şekil 12'de de görüleceği gibi, belirsiz bir komut verildiğinde, model istenen bilginin tamamını içermeyen yüzeysel bir yanıt üretebilir.

Model, belgeyi sadece düzeltcek mi yoksa belirli bir formatta düzenleyerek daha anlamlı hale getirebilir mi?

Yeni Sohbet

Kullanıcı

Yanlış Örnek

BİLGEM Yapay Zeka Asistanı

Lütfen düzeltmek istediğiniz metni sağlayın.

Metni düzelt.

Doğru Örnek

BİLGEM Yapay Zeka Asistanı

Dilekçe

Konu: Üniversite Kütüphanesine Yeni Bilgisayarların Sağlanması

Tarih: 15 Mart 2023

Hazırlayan: Ayşe Yılmaz

1. Giriş

Şekil 12. Talimat Temelli İstemleme

**Talimat temelli istemler**, modelin yönlendirilmesini kolaylaştırarak çıktıının doğruluğunu ve bağlam uygunluğunu artıran önemli bir tekniktir. **Açık ve yapılandırılmış bir istem** kullanıldığında, model daha anlamlı, tutarlı ve belirlenen formatta çıktı oluşturabilir.

### 3. Örnek Destekli İstemleme (Example-Based Prompting)

Örnek destekli istemleme yöntemi, modelin **belirli bir formatta, üsluba veya yapıda yanıt üretmesini sağlamak için** önceden sunulan örnekleri temel almasına dayanır. Bu teknik, **belirli bir yazım tarzının, dil seviyesinin veya içerik düzeninin korunması gereken** durumlarda oldukça etkilidir.

Şekil 13'te de görüleceği gibi, model örnek verilmeden yönlendirildiğinde, belirli bir üsluba veya yapıya bağlı kalmadan serbest bir yanıt üretebilir.

The screenshot shows a conversational interface between a user ('Kullanıcı') and a virtual assistant ('BİLGEM Yapay Zeka Asistanı').

**Top Panel:**

- User message: "Model, motivasyon konuşmasını rastgele mi oluşturacak yoksa belirli bir üslup ve örnek üzerinden mi şekillendirecek?"
- Assistant response: "Bir motivasyon konuşması yaz."

**Left Column (Yanlış Örnek):**

- Icon: A red circle with a white 'X' inside.
- Text: "Yanlış Örnek"
- Message: "BİLGEM Yapay Zeka Asistanı Uyanış Çağrısı" followed by a text block: "İçinizde bir kılçım var. Belki şimdi ışık yanıyor, belki de tamamen sönmüş gibi hissediyor. Ama unutmayın, her bireyin içinde bir güç kaynağı yatıyor. Siz, bu gücün farkında olmayı, onu harekete geçirmeyi bekleyen potansiyelsiniz."

**Right Column (Doğru Örnek):**

- Icon: A green circle with a white checkmark inside.
- Text: "Doğru Örnek"
- Message: "Aşağıdaki örnek gibi bir motivasyon konuşması hazırla: Örnek: 'Hayat, cesaretle ilerleyenleri ödüllendirir. Başarsızlık korkusunu yemek, büyük başarının anahtarıdır.' Benzer bir üslupla, ilham verici bir motivasyon konuşması yaz."
- Message: "BİLGEM Yapay Zeka Asistanı" followed by a text block: "Hedeflerinize ulaşmak için tek bir yol vardır: Vazgeçmemek. Zorluklar karşısında yılmayın, engeller karşısında pes etmeyin. Çünkü her zorluk, sizi daha güçlü kılan bir basamaktır. Her başarısızlık siz hedefinize bir adım daha"

**Şekil 13. Örnek Destekli İstemleme**

Örnek destekli istemleme, iyi seçilmiş bir örnek içeren bir istem kullanıldığında, modelin verilen örneğe uygun bir üslup ve yapı benimseyerek daha tutarlı, amaca yönelik ve istenen formatta yanıt üretmesini sağlar.

Bu yöntem, modelin beklentilere daha iyi uyum sağlamasını ve istenen standartlarda çıktı oluşturmasını sağlayan güçlü bir tekniktir.

## 4. Düşünce Zinciri İstemleme (CoT)

Düşünce zinciri yöntemi, modelin karmaşık problemleri aşamalara ayırarak çözmesini sağlayan bir tekniktir. Bu yaklaşım, problemi adım adım ele alarak daha mantıklı ve tutarlı sonuçlar elde etmeye yardımcı olur.

Özellikle büyük veri analizi, matematiksel işlemler ve mantıksal akıl yürütme gerektiren senaryolarda, tek bir yanıt üretmek yerine ara adımlar belirlemek modelin daha sistemli çalışmasını sağlar.

Şekil 14'te de görülebileceği gibi, modelin doğrudan büyük bir veri kümesi üzerinde analiz yapması durumunda, çıktı rastgele veya eksik olabilir.

The screenshot shows a conversation between a user and a large language model (LLM). The user asks the LLM to analyze a large dataset, but the LLM's response is random and irrelevant. This is labeled as an 'incorrect example' (Yanlış Örnek).

**Yanlış Örnek:**

Model, belgeyi sadece düzeltcek mi yoksa belirli bir formatta düzenleyerek daha anlamlı hale getirebilir mi?

**Kullanıcı:**

Yeni Sohbet

**Yanlış Örnek:**

Bana bu veri kümesiyle analiz yap.

**Doğru Örnek:**

Aşağıdaki büyük veri analiz problemini böл ve fethet yöntemiyle çöz:  
Problem: 1 milyon satırlık bir e-ticaret verisinde hangi ürünlerin en çok tercih edildiğini belirlemek istiyoruz.  
Çözüm Adımları:  
Veriyi kategori bazında alt kümelere ayır.  
Her kategori içinde en çok satılan ilk 10 ürünü belirle. Tüm kategorilerden gelen sonuçları birleştir ve genel eğitimleri analiz et.  
Sonuçları görselleştirerek bir rapor oluştur.  
Bu adımları takip ederek büyük ölçekli veri analizini daha yönetilebilir hale getir. Şimdi ilk aşama ile başlayalım.

**BİLGEM Yapay Zeka Asistanı:**  
Harika! Böl ve fethet yöntemiyle bu büyük veri analizini daha kolay yönetebiliriz.

**Şekil 14. Düşünce Zinciri İstemleme (CoT)**

Düşünce Zinciri İstemleme, problem aşamalara ayrıldığında, modelin her adımı ayrı ayrı ele alarak daha düzenli ve güvenilir bir analiz yapmasını sağlar.

Bu yöntem, modelin karmaşık görevleri daha verimli bir şekilde tamamlamasına yardımcı olan etkili bir tekniktir.

## 5. Düşünce Programlama İstemleme (PoT)

Bu yöntem, dil modelinin **hesaplama tabanlı bir akıl yürütme** süreci izlemesini sağlayarak, kod yazma ve mantıksal problem çözme gerektiren görevlerde daha **yapilandırılmış yanıtlar** üretmesine yardımcı olur. Modelden yalnızca bir kod çıktısı üretmesi değil, aynı zamanda bu sonuca nasıl ulaştığını da programatik olarak açıklaması beklenir.

Şekil 15'te de görülebileceği gibi, yanlış örnekte model, istenilen kodu nasıl yazacağı konusunda belirsizlik yaşamakta ve çıktıyı yeterince açıklayamamaktadır.

The screenshot shows a conversation between a user and a BİLGEM Yapay Zeka Asistanı (BİLGEM Artificial Intelligence Assistant).

**User Message:** Model, istedğim kodu belirlediğim dilde üretmesini ve çözüm sürecini adım adım açıklamasını nasıl sağlarım?

**Assistant Response:** Çift sayıları filtreleme kodu yaz.

**User Message:** Elbette, çift sayıları filtreleyen bir kod örneği yazabilirim. Hangi programlama dilini kullanmayı tercih edersiniz?

**Assistant Response:** Python kullanarak, verilen bir listenin içindeki çift sayıları filtreleyen bir fonksiyon yaz ve kodun nasıl çalıştığını açıkla.

**Code Snippet:**

```

Copy Code

def cift_sayıları_filtrele(sayılar):
    """
    Verilen bir listedeki çift sayıları filtreler.

    Args:
        sayılar: Bir tam sayı listesi.
    """

```

**Yanlış Örnek:** A message with a red 'X' icon indicating an incorrect example.

**Doğru Örnek:** A message with a green checkmark icon indicating a correct example.

Şekil 15. Düşünce Programlama İstemleme (PoT)

Bu tür durumlarda, model yalnızca bir kod çıktısı oluşturabilir, ancak kodun nasıl çalıştığını ve hangi mantıkla yazıldığını açıklamaz. Bu durum, özellikle programlamaya yeni başlayan kullanıcılar için kodun anlaşılmasını zorlaştırabilir.

Ancak, doğru örnekte model yalnızca kod üretmekle kalmayıp, aynı zamanda kodun işleyişini detaylandırarak açıklamaktadır. Programlama Tabanlı Düşünce Modelleme (PoT) yöntemi, modelin **kod üretirken daha öğretici ve sistematik bir yaklaşım benimsemesini sağlayarak**, kullanıcıya daha kapsamlı bir deneyim sunar.

## 6. Çok Adımlı İstemleme (Multi-Turn Prompting)

Çok adımlı istem yöntemi, modelle etkileşimli bir süreç oluşturarak, tek seferde kapsamlı bir yanıt almak yerine aşama aşama ilerleyerek daha detaylı ve yapılandırılmış çıktılar üretmeyi amaçlar. Bu yöntemde, modelin yönlendirilmesi adım adım gerçekleştiği için yanıtlar daha net, kapsamlı ve amaca uygun hale gelir.

Bu yaklaşım, Düşünce Zinciri (CoT) yönteminden farklı olarak, modelin kendi başına mantıksal bir akıl yürütme süreci oluşturmaya çalışmasını değil, kullanıcının gelen yönlendirmelere göre içeriği aşamalı olarak oluşturmasını sağlar. Özellikle uzun ve detaylı içerik gerektiren durumlarda, örneğin **ürün tanıtımları, rapor yazımı, eğitim materyalleri veya proje planları gibi** aşamalı anlatımların gerekligi senaryolarda oldukça etkilidir.

Şekil 16'da da görülebileceği gibi, yanlış örnekte model yüzeysel ve belirsiz bir yanıt üretmekte ve kullanıcının ek bilgi istemektedir.

**Yanlış Örnek**

Modelin, tek seferde yüzeysel bir yanıt vermesi yerine, daha detaylı ve yönlendirilmiş bir içerik oluşturmasını nasıl sağlayabilirim?

**Yeni Sohbet**

Kullanıcı

**BİLGEM Yapay Zeka Asistanı**

Harika! Daha spesifik bilgi vermenizi rica ediyorum. Örneğin:

- Hangi ürünü tanıtmak istiyorsunuz? (Bir uygulama mı, fiziksel bir ürün mü, hizmet mi?)
- Hedef kitleniz kim? (Gençler mi, profesyoneller mi, özel bir ilgi alanına sahip kişiler mi?)
- Ürününüzün en önemli özellikleri neler? (Rakiplerinden ayıran özellikler neler?)

**Doğru Örnek**

Yeni bir akıllı saat için ürün tanıtımı oluştur. Önce teknik özelliklerini açıkla. Ardından kullanım alanlarını belirt. Son olarak, bu ürüne uygun bir pazarlama sloganı oluştur.

**BİLGEM Yapay Zeka Asistanı**

**Zamanınızı Yeniden Keşfedin: Yeni “Chronos” Akıllı Saat**

Teknoloji ve Stilin Mükemmeli Birleşimi:

Chronos, son teknoloji ile donatılmış, şık ve kullanımı kolay bir akıllı saatidir.

Teknik Özellikler:

- Geniş AMOLED ekran:** Net ve canlı görüntü kalitesi.

Şekil 16. Çok Adımlı İstemleme

**Çok Adımlı İstemleme**, çıktıının doğrudan kullanılmasını zorlaştıracak ek etkileşim gerektirmesine yol açabilir. Ancak, doğru örnekte model, çok adımlı yönlendirmelerle hareket ederek belirlenen konuyu daha kapsamlı ve hedefe yönelik bir şekilde ele almıştır.

Bu yöntem, modelin aşama aşama içerik üretmesini sağlayarak daha detaylı ve yapısal çıktılar sunmasını mümkün kılar. **Özellikle uzun ve açıklayıcı yanıtlar gerektiren durumlarda, çok adımlı istemleme**, model çıktılarının doğruluğunu ve kullanım kolaylığını artıran önemli bir araçtır.

## 7. Rol Tabanlı İstemleme (Role-Based Prompting)

Rol tabanlı istemleme yöntemi, modelin belirli bir uzmanlık alanı veya kimlik doğrultusunda yanıt vermesini sağlamak için kullanılır. Model, verilen rol çerçevesinde yönlendirildiğinde daha tutarlı, detaylı ve uzmanlık gerektiren yanıtlar üretebilir.

Bu yöntem; finans, teknik danışmanlık, eğitim, mühendislik gibi belirli bilgi alanlarında modelin uzman bir perspektiften yanıt vermesine olanak tanır. Model, belirlenen role uygun şekilde terimler kullanarak, konuya özgü ve daha profesyonel bir üslup benimseyebilir.

Şekil 17'de de görülebileceği gibi, yanlış örnekte model, finans hakkında genel bir tanım yapmakta ve kullanıcının ihtiyacına uygun detaylı bir yanıt sunamamaktadır.

Modelin, belirli bir uzman rolü üstlenerken daha spesifik ve profesyonel yanıtlar vermesini nasıl sağlayabilirim?

**Yeni Sohbet**

**Yanlış Örnek**

**Doğru Örnek**

Bana finans hakkında bilgi ver.

**BİLGEM Yapay Zeka Asistanı**

Finans, paranın yönetimi, yatırımı ve kullanımı ile ilgili geniş bir alandır. Kişisel finans, şirket finansı ve kamu finansı gibi birçok alt dalı vardır.

Sen faydalı bir finansal danışmanskın. 5 yıllık yatırım stratejisi öner. Riskleri ve potansiyel getirileri değerlendir.

**BİLGEM Yapay Zeka Asistanı**

**5 Yıllık Yatırım Stratejisi: Riskler ve Getiriler**

- 1. Yatırım Hedefini ve Risk Toleransını Belirle
  - Düşük risk: Tahviller, temettü hisseleri
  - Orta risk: Endeks fonları, GYO
  - Yüksek risk: Teknoloji hisseleri, kripto
- 2. Portföy Çeşitlendirmesi
  - %30-40: Güvenli yatırımlar (tahviller, mevduat)
  - %30-40: Dengeli yatırımlar (ETF, gayrimenkul)

Şekil 17. Rol Tabanlı İstemleme

Modelin belirli bir uzmanlık bakış açısından yönlendirilmemesi, çıktıının yüzeysel kalmasına neden olmuştur. Ancak, doğruörnekte model bir finansal danışman rolüne yönlendirilmiş ve bu doğrultuda 5 yıllık yatırım stratejisi sunmuştur. Bu sayede, **model daha bağlama uygun**, kapsamlı ve profesyonel bir yanıt üretebilmiştir.

Rol tabanlı istemler, modelin daha doğru, uzmanlaşmış ve profesyonel çıktılar üretmesini sağlayarak, kullanıcının bekłentilerine daha uygun yanıtlar almasına yardımcı olan etkili bir tekniktir.

## 8. Kısıtlama Temelli İstemleme (Constraint-Based Prompting)

Kısıtlama temelli istem yöntemi, modelin belirlenen kriterlere uygun şekilde yanıt üretmesini sağlamak için kullanılır. Bu yaklaşım modelin serbestçe yanıt vermesi yerine, belirli çerçevede yönlendirilmesini gerektirir. Kelime sayısı sınırlamaları, **belirli veri formatlarında çıktı üretme** (örneğin, JSON, CSV, Markdown) veya dil bilgisi kurallarına uygunluk gibi kriterler bu yöntemin temel unsurlandandır.

Şekil 18'de de görülebileceği gibi, yanlışörnekte modelden JSON formatında bir çıktı üretmesi beklenmiş, ancak yeterli bağlam ve kısıtlamalar sağlanmadığı için sonuç eksik ve düzensiz olmuştur.

**Yanlış Örnek**

Modelden belirli bir formatta ve yapılandırılmış veri çıktıları almayı nasıl sağlayabilirim?

**BİLGEM Yapay Zeka Asistanı**

Copy Code

```
#İhtiyaç duyulan değişken isimlerine yer verilmemiştir bu örnekte.
```

**Doğru Örnek**

Bana JSON formatında ürün kategorileri için dummy veri ver.

**BİLGEM Yapay Zeka Asistanı**

Copy Code

```
{
  "id": 1,
  "name": "Ürün Adı",
  "category": "Kategori",
  "price": 100.50,
  "stock": 20
}
```

Aşağıdaki JSON iskeletine uygun olarak 10 farklı veri üret:  
Veriler en az üç farklı kategoriye ait olmalı (örneğin, elektronik, giyim, kitap).  
Fiyatlar 10 ile 1000 arasında rastgele belirlenmelidir.  
Stok miktarları ise 1 ile 100 arasında rastgele atanmalıdır.

Şekil 18. Kısıtlama Temelli İstemleme

Model, değişken isimleri ve veri yapısını belirli bir düzene oturtmadığı için kullanışlı olmayan bir çıktı üretmiştir.

Buna karşılık, doğru örnekte belirli bir JSON şeması, kategori türleri ve rastgele değer atama koşulları net bir şekilde tanımlanmıştır. Bu sayede model, kullanıcı gereksinimlerine uygun, yapılandırılmış ve tekrar kullanılabilir bir çıktı oluşturmuştur.

Kısıtlama temelli istemler, modelin belirlenen formatlara ve kurallara bağlı kalarak daha tutarlı, güvenilir ve kullanışlı yanıtlar üretmesini sağlayan etkili bir tekniktir. Özellikle veri analizi, kod üretime ve yapılandırılmış içerik oluşturma gibi görevlerde büyük avantaj sağlar.

## 9. Hafıza Destekli İstemleme (Memory-Aware Prompting)

Hafıza destekli istemleme, modelin önceki girdileri veya konuşmaları hatırlamasını sağlayarak daha tutarlı ve anlamlı yanıtlar üretmesine yardımcı olur. Özellikle uzun süreli diyaloglar, devam eden analizler ve bağlamın korunmasının önemli olduğu senaryolarda etkili bir şekilde kullanılır.

Şekil 19'da da görülebileceği gibi, yanlış örnekte model önceki konuşmaları hatırlayamadığı için bağlamı kaybetmiş ve yeterince tatmin edici bir yanıt üretmemiştir.

Model, önceki konuşmaları nasıl hatırlayabilir ve yanıtlarını önceki bağlamla nasıl uyumlu hale getirebilir?

?
Kullanıcı

X

Yanlış Örnek

Daha önce konuştuğumuz konu hakkında bilgi ver.

**BİLGEM Yapay Zeka Asistanı**

Üzgünüm, ancak geçmiş konuşmalarımı hatırlamıyorum.

✓

Doğru Örnek

Önceki konuşmamıza referansla, yapay zeka etik kurallarına dair özet çıkar.

**BİLGEM Yapay Zeka Asistanı**

**Yapay Zeka Etik Kuralları-Önceki Konuşmaya Referansla Özeti**

Önceki konuşmamıza dayanarak, yapay zeka etik kuralları aşağıdaki başlıklar çerçevesinde ele alınabilir:

1. **Veri Gizliliği ve Güvenlik:** Yapay zeka sistemleri, kullanıcı verilerini işlerken mahremiyeti korumalı ve yetkisiz erişime karşı güvenlik önlemleri almmalıdır. Kişisel verilerin işlenmesi, yasal düzenlemelere uygun olmalıdır.
2. **Tarafsızlık ve Adalet:** Yapay zeka modellerinin yanlışlığı en aza indirilmeli ve algoritmaların her kesime eşit şekilde hizmet etmesi sağlanmalıdır. Veri

**Şekil 19. Hafıza Destekli İstemleme**

Sonuç olarak, kullanıcıya ilgili konuya ilgili bütüncül bir bilgi sunamamıştır. Ancak, doğru örnekte model önceki konuşmaya referans vererek bağlamı koruyabilmiş ve daha bütünlüklü bir yanıt oluş-

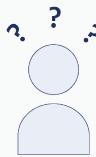
turmuştur. Hafıza destekli istemler, modelin geçmiş konuşmalarla bağlantılı, tutarlı ve konuya uyumlu çıktılar üretmesini sağlayarak kullanıcı deneyimini daha verimli hale getiren bir tekniktir.

## 10. Belirsizlik Azaltıcı İstemleme (Disambiguation-Based Prompting)

Belirsizlik azaltıcı istemleme, modelin yanlış anlamasını veya eksik yanıtlar vermesini önlemek için belirsiz ifadeleri netleştiren teknikleri kapsar. Genel veya çok anlamlı talepler, modelin doğru ve beklenen yanıt üretmesini zorlaştırabilir. Bu yöntemde, bağlamın güçlendirilmesi ve ek bilgiler eklenerek netlik sağlanması hedeflenir.

Şekil 20'de de görülebileceği gibi, belirsiz bir talep karşısında model yalnızca yüzeysel bir yanıt üretbilir ve kullanıcıya beklenen detayları sunamayabilir.

Modelin bağlam eksikliği nedeniyle yanlış veya eksik yanıtlar üretmesini nasıl önleyebilirim?



Kullanıcı

Yeni Sohbet

 Yanlış Örnek

**BİLGEM Yapay Zeka Asistanı**  
Python, 1991 yılında Guido van Rossum tarafından yaratılan, yüksek seviyeli, yorumlanan ve genel amaçlı bir programlama dilidir.

 Doğru Örnek

**BİLGEM Yapay Zeka Asistanı**  
Python, okunabilirliği ve basitliğiyle bilinen, çok amaçlı bir programlama dilidir.

**Kullanım Alanları:**

- Web geliştirme (Django, Flask)
- Veri bilimi ve makine öğrenmesi ( Pandas, NumPy, Scikit-learn)
- Scripting ve otomasyon
- Masaüstü uygulamaları

**Popüler Kütüphaneler:**

- **NumPy:** Sayısal hesaplama
- **Pandas:** Veri analizi ve manipülasyonu

Bana Python hakkında bilgi ver.

Python programlama dili hakkında temel bilgiler ver. Dilin kullanım alanlarını ve en popüler kütüphanelerini açıkla.  
Kısa ve öz

Şekil 20. Belirsizlik Azaltıcı İstemleme

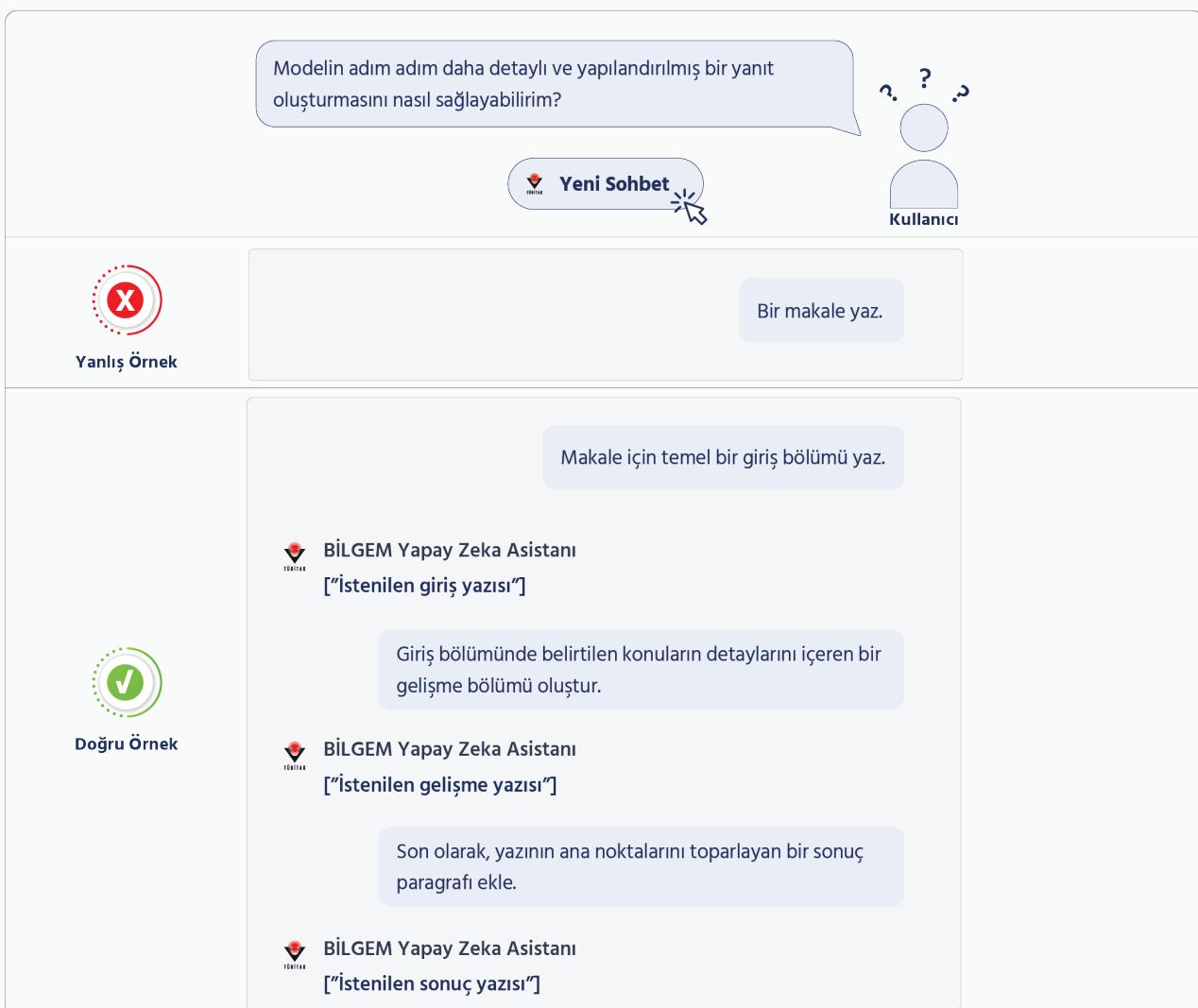
**Belirsizlik azaltıcı istemleme**, daha yönlendirilmiş bir istem kullanıldığında, modelin verdiği yanıtın daha kapsamlı ve konuya uygun hale gelmesini sağlar. Örneğin, yalnızca genel bir tanım yapmak yerine, kullanım alanları ve ilgili kütüphaneler gibi daha spesifik bilgiler içeren bir çıktı üretebilir.

Bu yöntem, modelin yanıtlarını daha doğru ve bağlama uygun hale getirerek, anlamlı ve detaylı çıktılar elde edilmesine yardımcı olur.

## 11. Aşamalı Zorluk İstemleme (Progressive Complexity Prompting)

Aşamalı zorluk istemleme yöntemi, modelin adım adım daha karmaşık görevleri yerine getirmesine olanak tanıyan bir istem teknigidir. İçerik üretiminin sistematik ve kontrollü bir şekilde ilerlemesini sağlar.

Bu yöntemde, modelden doğrudan geniş kapsamlı bir yanıt üretmesi beklenmez. Bunun yerine, aşamalı bir süreç izlenerek yanıtın her aşamada daha fazla detaylandırılması sağlanır. Şekil 21'de de görülebileceği gibi, yanlışörnekte model doğrudan uzun bir makale üretmeye çalışmış, ancak çıktı eksik ve yüzeysel kalmıştır.



Şekil 21. Aşamalı Zorluk İstemleme

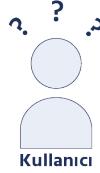
Belirli bir yapı olmadan üretilen bu tür yanıtlar, konunun kapsamlı şekilde ele alınmasını engelleyebilir. Ancak, doğruörnekte model, **aşamalı bir süreç** izleyerek önce temel çerçeveyi oluşturmuş, ardından her aşamada ek detaylar ekleyerek daha yapısal ve kapsamlı bir içerik üretmiştir. Bu teknik, modelin daha sistemli ve tutarlı bir içerik oluşturmmasına yardımcı olurken, aynı zamanda kullanıcıya daha anlaşılır ve derinlemesine bilgiler sunar.

## 12. Açıklamalı İstemleme (Annotation-Based Prompting)

Açıklamalı istemleme, modelin yanıtlarını daha yönlendirilmiş hale getirmek için ek ipuçları ve bağlama bilgiler içeren bir yöntemdir. Modelin belirli kriterlere uygun çıktılar üretmesini sağlamak için format ve bağlam net bir şekilde belirtilir.

Şekil 22'de görülebileceği gibi, yanlışörnekte model, yeterli bağlam ve format belirtilmeden yönlendirildiği için genel bir yanıt üretmiştir. Bu tür istemler, modelin beklenen bilgi seviyesine ulaşmasını zorlaştırabilir ve çıktıının amaca uygunluğunu azaltabilir.

**Yanlış Örnek:** Modelin çıktısını belirli bir format ve içerik düzenebine göre nasıl yönlendirebilirim?

**Kullanıcı:**  ?

**Yeni Sohbet:** 

**BİLGEM Yapay Zeka Asistanı:** Dosya form.pdf 120.0 KB  
Metni özetle.

**Doğru Örnek:** 

**BİLGEM Yapay Zeka Asistanı:** Dosya 2001.04451v2.pdf 643,1 KB  
Metnin temel fikirleini koruyarak, akademik bir üslupla 100 kelimeyle bir özeti çıkar. Özeten sonunda, metindeki en önemli çıkarımları listeleyin.

**BİLGEM Yapay Zeka Asistanı:** Bu metin, büyük dil modellerinin (LLM) yeteneklerini ve sınırlarını inceliyor. Özellikle, LLM'lerin bağlama duyarlılığı, bilgi temsili ve akıl yürütme yetenekleri üzerinde duruluyor. Metin, LLM'lerin dil modellemesindeki ilerlemeleri ve bu modellerin doğal dil işleme görevlerindeki başarısını vurgularken, aynı zamanda bu modellerin hala karşılaştığı zorlukları da ele alıyor.

**Önemli çıkarımlar:**

- LLM'ler, dil anlaması ve üretme konularında önemli

Şekil 22. Açıklamalı İstemleme

Bu teknik, modelin beklenen çıktıyı oluştururken daha kontrollü ve yapılandırılmış bir yaklaşım benimsemesini sağlar. Açıklamalar ve ipuçları eklendiğinde, model yalnızca genel bir yanıt vermek yerine, belirli bir formatı takip ederek daha organize ve amaca uygun sonuçlar üretebilir.

Doğruörnekte ise model, ek açıklamalar ve belirlenmiş bir format ile yönlendirildiğinde, daha yapılandırılmış, bağlama olarak tutarlı ve istenen kriterlere uygun bir yanıt üretmiştir.

# Gelecekteki Araştırma Yönetimleri

Büyük dil modelleri, son yıllarda dil işleme alanında önemli ilerlemeler kaydetmiş olsa da hâlâ gelişirmesi gereken birçok yönü bulunmaktadır. Özellikle istem tasarımları, model eğitimi ve mevcut sınırlamaları aşmaya yönelik yeni yaklaşımalar, bu alandaki araştırmaların odak noktaları olmaya devam etmektedir.

Gelecekte, dil modellerinin **bağlama duyarlı ve esnek istemlere** daha iyi yanıt verebilmesi için bu alanda kapsamlı çalışmalar yürütülmesi gerekmektedir. Bir modelin sorulan soruya tutarlı ve anlamlı bir yanıt üretmesi, yalnızca büyük veri kümeleriyle eğitilmiş olmasına değil, aynı zamanda bu veriyi **etkili şekilde işlemesine** bağlıdır. Bu nedenle, Düşünce Zinciri [Chain-of-Thought (CoT)] gibi tekniklerin **daha karmaşık problemlerin çözümüne nasıl entegre edilebileceği** üzerine yapılan araştırmalar önem taşımaktadır. Ayrıca, **kullanıcı girdilerine dinamik olarak uyarlanabilen istem yapıları** geliştirilerek modellerin bağlamsal anlayışının iyileştirilmesi hedeflenmektedir.

Büyük dil modellerinin eğitimi de araştırmalarda önemli bir yere sahiptir. Bu modeller, büyük veri kümeleri ve yüksek işlem gücü gerektirdiğinden, **eğitim süreçlerinin daha verimli hale getirilmesi** büyük bir öncelik oluşturmaktadır. **Transfer öğrenme [Transfer Learning]** ve **uzman karışımı [Mixture of Experts (MoE)]** gibi yöntemler sayesinde, **daha az veri kullanarak daha güçlü ve verimli modeller eğitmek mümkün hale gelmektedir**. Bunun yanı sıra, düşük kaynaklı dillerin daha iyi desteklenmesi için özelleştirilmiş eğitim süreçlerine ihtiyaç duyulmaktadır. Örneğin, **bölgesel diller için optimize edilmiş modellerin geliştirilmesi**, yapay zekanın küresel erişilebilirliğini artırabilir.

Dil modellerinin gelişimini engelleyen en büyük zorluklardan biri de **yanlılık ve etik sorunlardır**. Modelin eğitildiği veri kümelerindeki önyargılar, çıktılara doğrudan yansıyabilir ve bu da taraflı veya hatalı bilgi üretimine yol açabilir. **Bu sorunu azaltmak için**, veri seçimi süreçlerinin titizlikle yönetilmesi ve modellerin çıktılarındaki yanlışlıklarını **erken aşamada tespit edebilen sistemler geliştirilmesi** gerekmektedir. Ayrıca, **metin, görsel ve ses gibi farklı veri türlerini aynı modelde işleyebilen çok modlu [multi-modal] sistemlerin yaygınlaştırılması**, dil modellerinin kullanım alanlarını genişletecektir. Özellikle tıbbi teşhis, güvenlik ve otomasyon gibi alanlarda, çok modlu modeller büyük avantaj sağlayacaktır.

Bunun yanı sıra, **kullanıcı geri bildirimlerinin model geliştirme süreçlerine daha aktif entegre edilmesi**, sistemlerin zaman içinde daha güvenilir hale gelmesine katkı sağlayacaktır. Yapay zekanın **daha bağımsız düşününebilen, enerji açısından verimli ve etik değerlere duyarlı hale getirilmesi**, gelecekteki araştırmaların temel öncelikleri arasında yer alacaktır.

Sonuç olarak, **büyük dil modellerinin etkinliğini artırmak için hem teknik hem de etik boyutta gelişim sağlanmalıdır**. Bu amaçla model eğitimi ve istem tasarımlına yönelik yeni yaklaşımalar, yapay zeka sistemlerinin **daha tutarlı, güvenilir ve geniş çapta uygulanabilir hale gelmesini** destekleyecektr.

# Sonuç ve Öneriler

---

Büyük dil modellerinde **etkili istem tasarıımı**, yapay zekanın gerçek dünya problemlerine uygun çözümler üretmesini sağlamak açısından kritik bir rol oynamaktadır. Bu araştırma serisinde, dil modellerinin çalışma prensipleri, performanslarını artırmaya yönelik teknikler, farklı kullanım alanları ve etik sorumlulukları ele alınmıştır. Ayrıca modellerin karşılaştığı zorluklar ve bunları aşmak için geliştirilen yöntemler değerlendirilmiştir.

Büyük dil modelleri, **bağlamı doğru yorumlayabilmek** için yalnızca geniş veri kümelerine değil, aynı zamanda etkili yönlendirmelere de ihtiyaç duyar. Bu yüzden, **açık, öz ve hedef odaklı istemler**, modelin daha tutarlı ve güvenilir çıktılar üretmesini sağlar. Kullanıcıların model ile etkili bir etkileşim kurabilmesi, yalnızca net bekentiler tanımlamasıyla değil, aynı zamanda modeli yönlendirecek stratejik istemler kullanmasıyla da doğrudan ilişkilidir. Örneğin, adım adım düşünme yöntemleri veya örnek tabanlı istemleme gibi teknikler, modelin daha mantıklı ve bağlam odaklı yanıtlar üretmesini sağlayabilir.

Büyük dil modellerinin **performansı**, kullanım alanına bağlı olarak değişiklik göstermektedir. Örneğin, doküman analizi ve metin özetleme gibi görevlerde bazı modeller daha başarılı olabilirken, kod üretimi veya hata ayıklama süreçlerinde farklı modellerin tercih edilmesi gerekebilir. Bu nedenle, modelin sınırlarını anlamak ve kullanım amacına en uygun şekilde yönlendirmek önemlidir.

**Güvenlik ve etik kullanım** da büyük dil modellerinin geliştirilmesinde dikkate alınması gereken kritik konular arasındadır. Modeller, yanlış içerik üretebilir veya kullanıcı girdilerine karşı manipüle edilebilir. Bu tür riskleri minimize etmek için güvenlik önlemleri alınmalı, yaniltıcı veya zararlı içeriklerin üretilmesini engelleyen **filtreleme mekanizmaları** geliştirilmelidir. Ayrıca yapay zekanın etik ilkeler doğrultusunda çalışmasını sağlamak adına eğitim süreçlerinde daha dengeli veri kümeleri kullanılmalı ve model çıktıları düzenli olarak gözden geçirilmelidir.

Önümüzdeki yıllarda, yapay zeka sistemlerinin yalnızca **daha güçlü ve akıllı hale getirilmesi** değil, aynı zamanda daha güvenilir, **verimli ve çevre dostu** olması da önemli bir araştırma konusu olacaktır. Bunu başarmak için, enerji tüketimini azaltan ve kaynakları daha verimli kullanan hafifletilmiş modellerin yaygınlaştırılması gerekmektedir. Bunun yanı sıra, kullanıcıların yapay zeka ile daha etkili etkileşim kurabilmesi için **eğitim materyalleri** geliştirilmeli ve **bilinçli kullanım** teşvik edilmelidir.

Sonuç olarak, **büyük dil modelleri**, yalnızca metin işleme süreçlerinde değil, aynı zamanda **yapay zekanın geniş çapta uygulanabilirliğini artırmada** da kritik bir rol oynamaktadır. **İstem mühendisliği**, modellerin potansiyelini en üst düzeye çıkarmak ve daha güvenilir, doğru ve etik sistemler geliştirmek için temel bir unsurdur. Yapay zekanın gelecekteki gelişimi, bu sistemlerin nasıl eğitildiği, yönlendirildiği ve etik sınırlar içinde nasıl optimize edildiğiyle doğrudan bağlantılı olacaktır.

# Kaynakça

---

- [1] J. Gui vd., "A Survey on Self-supervised Learning: Algorithms, Applications, and Future Trends", Oca. 2023, [Çevrim içi]. Erişim adresi: <http://arxiv.org/abs/2301.05712>.
- [2] D. E. Rumelhart, G. E. Hinton, ve R. J. Williams, "Learning representations by back-propagating errors", Nature 1986 323:6088, c. 323, sy 6088, ss. 533-536, 1986, doi: 10.1038/323533a0.
- [3] F. Rosenblatt, "THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN", Psychol Rev, c. 65, sy 6, ss. 19-27, 1958.
- [4] Y. Bengio, J. Louradour, R. Collobert, ve J. Weston, "Curriculum learning", ACM International Conference Proceeding Series, c. 382, 2009, doi: 10.1145/1553374.1553380.
- [5] J. L. Elman, "Finding Structure in Time", Cogn Sci, c. 14, sy 2, ss. 179-211, Mar. 1990, doi: 10.1207/S15516709COG1402\_1.
- [6] R. Pascanu, T. Mikolov, ve Y. Bengio, "On the difficulty of training Recurrent Neural Networks", 30th International Conference on Machine Learning, ICML 2013, sy PART 3, ss. 2347-2355, Kas. 2012, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/1211.5063v2>.
- [7] S. Hochreiter ve J. Schmidhuber, "Long Short-Term Memory", Neural Comput, c. 9, sy 8, ss. 1735-1780, Kas. 1997, doi: 10.1162/NECO.1997.9.8.1735.
- [8] K. Cho vd., "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation", EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, ss. 1724-1734, Haz. 2014, doi: 10.3115/v1/d14-1179.
- [9] A. Vaswani vd., "Attention Is All You Need", Adv Neural Inf Process Syst, c. 2017-December, ss. 5999-6009, Haz. 2017, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/1706.03762v7>.
- [10] "Compare the different Sequence models (RNN, LSTM, GRU, and Transformers) - AIML.com". Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://aiml.com/compare-the-different-sequence-models-rnn-lstm-gru-and-transformers/>.
- [11] M. O. Topal, A. Bas, ve I. van Heerden, "Exploring Transformers in Natural Language Generation: GPT, BERT, and XLNet", Şub. 2021, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/2102.08036v1>.

- [12] T. B. Brown vd., "Language Models are Few-Shot Learners", *Adv Neural Inf Process Syst*, c. 2020-December, May. 2020, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/2005.14165v4>.
- [13] J. Devlin, M. W. Chang, K. Lee, ve K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, c. 1, ss. 4171-4186, Eki. 2018, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/1810.04805v2>.
- [14] DeepSeek-AI vd., "DeepSeek-V3 Technical Report", Ara. 2024, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/2412.19437v2>.
- [15] DeepSeek-AI vd., "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning", Oca. 2025, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/2501.12948v1>.
- [16] S. Zhang vd., "Instruction Tuning for Large Language Models: A Survey", Agu. 2023, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/2308.10792v8>.
- [17] L. Ouyang vd., "Training language models to follow instructions with human feedback", *Adv Neural Inf Process Syst*, c. 35, Mar. 2022, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/2203.02155v1>.
- [18] S. Schulhoff vd., "The Prompt Report: A Systematic Survey of Prompt Engineering Techniques", Haz. 2024, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/2406.06608v6>.
- [19] I. Y. Tyukin, A. N. Gorban, M. H. Alkhudaydi, ve Q. Zhou, "Demystification of Few-shot and One-shot Learning", *Proceedings of the International Joint Conference on Neural Networks*, c. 2021-July, Tem. 2021, doi: 10.1109/IJCNN52387.2021.9534395.
- [20] J. Wei vd., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models", *Adv Neural Inf Process Syst*, c. 35, Oca. 2022, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/2201.11903v6>.
- [21] M. Hewing ve V. Leinhos, "The Prompt Canvas: A Literature-Based Practitioner Guide for Creating Effective Prompts in Large Language Models", Ara. 2024, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/2412.05127v1>.
- [22] B. Lester, R. Al-Rfou, ve N. Constant, "The Power of Scale for Parameter-Efficient Prompt Tuning", *EMNLP 2021 - 2021 Conference on Empirical Methods in Natural Language Processing, Proceedings*, ss. 3045-3059, Nis. 2021, doi: 10.18653/v1/2021.emnlp-main.243.

- [23] M. Besta vd., "Graph of Thoughts: Solving Elaborate Problems with Large Language Models", Proceedings of the AAAI Conference on Artificial Intelligence, c. 38, sy 16, ss. 17682-17690, Agu. 2023, doi: 10.1609/aaai.v38i16.29720 .
- [24] S. Yao vd., "Tree of Thoughts: Deliberate Problem Solving with Large Language Models", Adv Neural Inf Process Syst, c. 36, May. 2023, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/2305.10601v2> .
- [25] Y. Yao, Z. Li, ve H. Zhao, "Beyond Chain-of-Thought, Effective Graph-of-Thought Reasoning in Language Models", May. 2023, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/2305.16582v2> .
- [26] W. Chen, X. Ma, X. Wang, ve W. W. Cohen, "Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks", Kas. 2022, Erişim: 15 Mart 2025. [Çevrim içi]. Erişim adresi: <https://arxiv.org/abs/2211.12588v4> .



İşçi Blokları Mahallesi Muhsin Yazıcıoğlu Caddesi No:51/C 06530 Çankaya/ANKARA

+90 (312) 289 92 22 - [YTE.Bilgi@tubitak.gov.tr](mailto:YTE.Bilgi@tubitak.gov.tr)

TÜBİTAK - BİLGE Yazılım Teknolojileri Araştırma Enstitüsü (YTE)