VIP Cheatsheet: Dönüştürücüler ve Büyük Dil Modelleri

Afshine Amidi ve Shervine Amidi
Merve Ayyüce Kızrak tarafından çevirildi

1 Nisan 2025

Bu VIP özet rehberi, 250 sayfa boyunca ∼600 resim içeren ve aşağıdaki kavramları derinlemesine ele alan "Super Study Guide: Dönüştürücüler & Büyük Dil Modelleri" kitabında bulunanlara genel bir bakış sunar. Daha fazla ayrıntıyı `https://superstudy.guide` *adresinde bulabilirsiniz.*

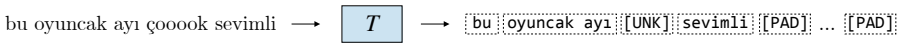
1 Temeller

1.1 Birimler

❒ **Tanım** – *Birim* (*token*), bir sözcük, alt sözcük veya karakter gibi bölünemez bir metin birimidir ve önceden tanımlanmış bir sözlüğün parçasıdır.

Not: Bilinmeyen birim `[UNK]` *bilinmeyen metin parçalarını temsil ederken, doldurma birimi* `[PAD]` *tutarlı giriş dizisi uzunluklarını sağlamak için boş pozisyonları doldurmak amacıyla kullanılır.*

❒ **Bölütleyici** – Bir *bölütleyici* (*tokenizer*) T, metni isteğe bağlı bir ayrıntı düzeyindeki birimlere böler.



Bölütleyicilerin başlıca tipleri şunlardır:

Tip	Artılar	Eksiler	Gösterim
Sözcük	• Yorumlanması kolay • Kısa dizi	• Büyük sözcük varlığı boyutu • Sözcük varyasyonları işlenmez	`oyuncak` `ayı`
Alt Sözcük	• Sözcük köklerinin kullanılması • Sezgisel sözcük temsilleri (embeddings)	• Artan dizi uzunluğu • Bölütleyici daha karmaşık	`oyun` `##cak` `ayı`
Karakter Bayt	• Söz varlığının dışında kalanlar için sorun yok • Küçük sözcük varlığı boyutu	• Çok daha uzun dizi uzunluğu • Çok düşük seviyeli oldukları için yorumlanması zor örüntüler	`o` `y` `u` `n` `c` `a` `k` ` ` `a` `y` `ı`

Not: Bayt-çifti kodlama (Byte-Pair Encoding, BPE) ve tek-gram (Unigram), yaygın olarak kullanılan alt sözcük düzeyindeki bölütleyicilerdir.

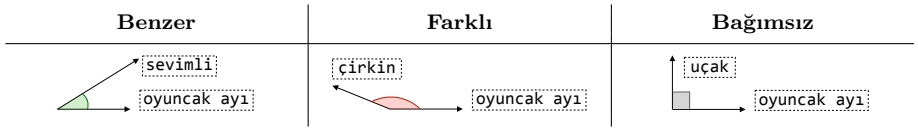
1.2 Sözcük temsilleri

❒ **Tanım** – Bir *sözcük temsili*, bir öğenin (örneğin, birim, cümle) sayısal bir gösterimidir ve $x \in \mathbb{R}^n$ vektörü ile karakterize edilir.

❒ **Benzerlik** – İki birim t_1, t_2 arasındaki *kosinüs benzerliği* şu şekilde ölçülür:

$$\boxed{\text{benzerlik}(t_1, t_2) = \frac{t_1 \cdot t_2}{||t_1||\ ||t_2||} = \cos(\theta) \in [-1, 1]}$$

θ açısı iki birim arasındaki benzerliği karakterize eder:

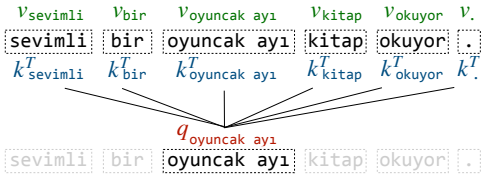


Not: Yaklaşık en yakın komşular (Approximate Nearest Neighbors, ANN) ve yerel-duyarlı kodlandırma (Locality-Sensitive Hashing, LSH), büyük veritabanları üzerinde benzerlik işlemini verimli bir şekilde yaklaşık olarak hesaplayan yöntemlerdir.

2 Dönüştürücüler

2.1 Dikkat

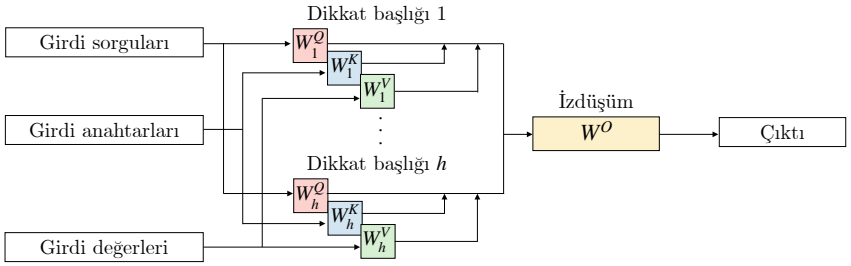
❒ **Formül** – Bir q *sorgusu* verildiğinde, sorgunun ilişkili *değer* v ile ilgili olarak hangi k *anahtarına* "dikkat" etmesi gerektiğini bilmek istiyoruz.



Dikkat, sırasıyla sorgular q, anahtarlar k ve değerler v içeren Q, K, V matrisleri ve anahtarların boyutu d_k kullanılarak verimli bir şekilde hesaplanabilir:

$$\boxed{\text{dikkat} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V}$$

❒ **MHA** – *Çoklu-başlıklı dikkat* (***M**ulti-**H**ead **A**ttention*, MHA) katmanı, birden fazla başlık arasında dikkat hesaplamaları gerçekleştirir ve ardından sonucu çıktı alanına yansıtır.

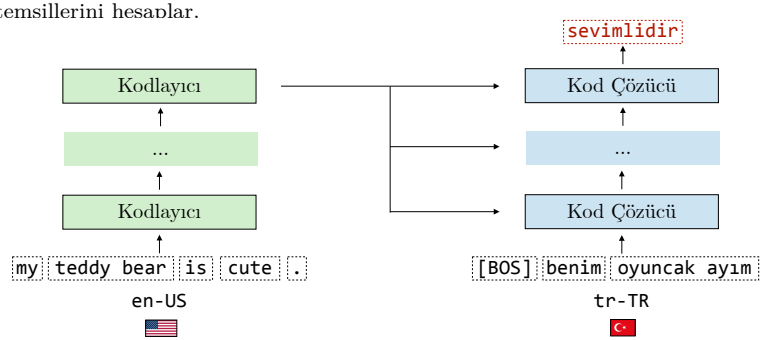


h dikkat başlığından ve Q sorgularını, K anahtarlarını ve V değerlerini elde etmek için girdinin izdüşümü olan W^Q, W^K, W^V matrislerinden oluşur. İzdüşüm W^O matrisi kullanılarak oluşturulur.

Not: Gruplu-sorgu dikkat (Grouped-Query Attention, GQA) ve çoklu-sorgulu dikkat (MQA), anahtarları ve değerleri dikkat başlıkları arasında paylaşarak hesaplama yükünü azaltan MHA varyasyonlarıdır.

2.2 Mimari

❒ **Genel bakış** – *Dönüştürücü* (*Transformer*), öz-dikkat mekanizmasına dayanan bir dönüm noktası niteliğinde modelidir ve kodlayıcılar ve kod çözücülerden oluşur. Kodlayıcılar, kod çözücüler tarafından dizideki bir sonraki birimii tahmin etmek için kullanılan girdinin anlamlı sözcük temsillerini hesaplar.



Not: Dönüştürücü başlangıçta çeviri görevleri için bir model olarak önerilmiş olsa da, günümüzde birçok başka uygulamada da yaygın olarak kullanılmaktadır.

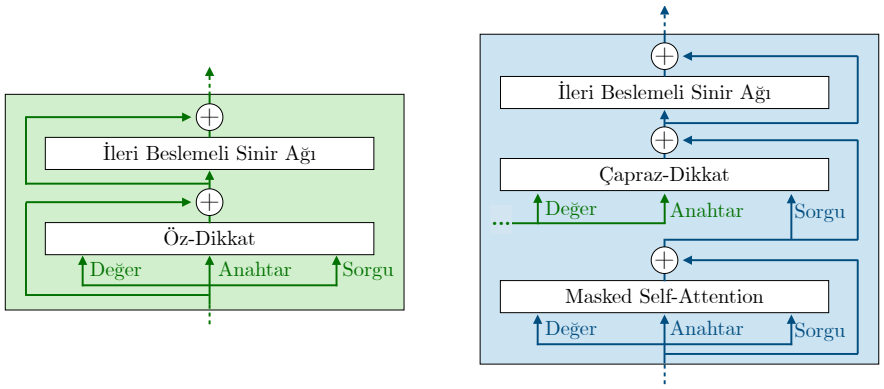
❒ **Bileşenler** – *Kodlayıcı* ve *kod çözücü*, Dönüştürücünün iki temel bileşenidir ve farklı rollere sahiptir:

Kodlayıcı

Kodlanmış sözcük temsilleri girdinin anlamını kapsül haline getirir

Kod çözücü

Kod çözücü uygulanmış sözcük temsilleri, şimdiye kadar tahmin edilen hem girdinin hem de çıktının anlamını kapsül haline getirir

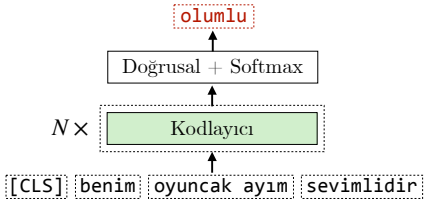


❒ **Konum sözcük temsili** – *Konum sözcük temsilleri*, birimin cümlede nerede olduğunu bildirir ve birim sözcük temsilleriyle aynı boyuttadır. Bunlar isteğe bağlı olarak tanımlanabilir veya verilerden öğrenilebilir.

Not: Dönüşlü konum sözcük temsilleri (Rotary Position Embeddings, RoPE), göreceli konum bilgilerini dahil etmek için sorgu ve anahtar vektörlerini döndüren popüler ve etkili bir varyasyondur.

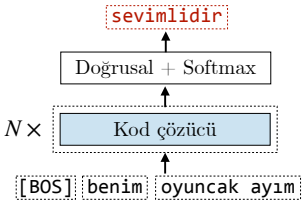
2.3 Türler

❒ **Yalnızca kodlayıcı** – *Dönüştürücülerden çift yönlü kodlayıcı gösterimleri* (***B**idirectional **E**ncoder **R**epresentations from **T**ransformers*, BERT), girdi olarak bir miktar metin alan ve daha sonra alt akış sınıflandırma görevlerinde kullanılabilen anlamlı sözcük temsilleri üreten bir kodlayıcı yığınından oluşan Dönüştürücü tabanlı bir modeldir.



Cümlenin anlamını yakalamak için dizinin başına bir `[CLS]` birimi eklenir. Kodlanmış sözcük temsili genellikle duygu çıkarma gibi alt akış görevlerinde kullanılır.

❒ **Yalnızca kod çözücü** – *Üretici önceden eğitilmiş dönüştürücü* (***G**enerative **P**re-trained **T**ransformer*, GPT), bir kod çözücü yığınından oluşan otoregresif Dönüştürücü tabanlı bir modeldir. BERT ve türevlerinin aksine, GPT tüm sorunları metinden metne sorunlar olarak ele alır.



Günümüzdeki en son BDM'lerin çoğu, GPT serisi, LLaMA, Mistral, Gemma, DeepSeek vb. gibi yalnızca kod çözücü mimarisine dayanmaktadır.

Not: T5 gibi kodlayıcı-kod çözücü modelleri de otoregresiftir ve yalnızca kod çözücü modellerle birçok özelliği paylaşır.

2.4 Optimizasyonlar

❒ **Dikkat yaklaşımı** – Dikkat hesaplamaları $\mathcal{O}(n^2)$'dedir ve bu, dizi uzunluğu n arttıkça maliyetli olabilir. Hesaplamaları yaklaşık olarak hesaplamak için iki ana yöntem vardır:

- *Seyreklik*: Öz-dikkat tüm dizi boyunca gerçekleşmez, yalnızca daha alakalı birimler arasında gerçekleşir.

- *Düşük dereceli*: Dikkat formülü, hesaplama yükünü azaltan düşük dereceli matrislerin çarpımı şeklinde basitleştirilir.

❒ **Ani dikkat** – *Ani dikkat* (*Flash attention*), GPU donanımını akıllıca kullanarak, matris işlemleri için hızlı *statik rastgele-erişimli belleği* (***S**tatic **R**andom-**A**ccess **M**emory*, SRAM) kullanarak ve sonuçları daha yavaş *yüksek bant genişlikli belleğine* (***H**igh **B**andwidth **M**emory*, HBM) yazmadan önce dikkat hesaplamalarını optimize eden kesin bir yöntemdir.

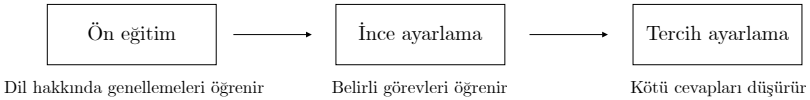
Not: Uygulamada, bu bellek kullanımını azaltır ve hesaplamaları hızlandırır.

3 Büyük dil modelleri

3.1 Genel bakış

❒ **Tanım** – ***B**üyük **D**il **M**odeli* (BDM), güçlü DDİ yeteneklerine sahip Dönüştürücü tabanlı bir modeldir. Genellikle milyarlarca parametre içermesi anlamında "büyük"tür.

❒ **Yaşam döngüsü** – BDM eğitimi 3 aşamada gerçekleşir: ön eğitim, ince ayar ve tercih ayarı.

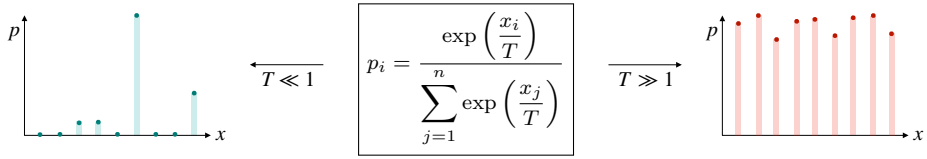


İnce ayarlama ve tercih ayarlaması, modeli belirli görevleri yerine getirecek şekilde hizalamayı amaçlayan eğitim sonrası yaklaşımlardır.

3.2 Komutlama

❒ **Bağlam uzunluğu** – Bir modelin bağlam uzunluğu, girdiye sığabilecek maksimum birim sayısıdır. Genellikle on binlerce ile milyonlarca birim arasında değişir.

❒ **Kod çözme örneklemesi** – Birim tahminleri, hiperparametre sıcaklığı T tarafından kontrol edilen tahmin edilen olasılık dağılımı p_i'den örneklenir.



Not: Yüksek sıcaklıklar daha yaratıcı çıktılara yol açarken, düşük sıcaklıklar daha kesin çıktılara yol açar.

❒ **Düşünce zinciri** – *Düşünce zinciri* (***C**hain-**o**f-**T**hought*, CoT), modelin karmaşık bir problemi bir dizi ara adıma böldüğü bir akıl yürütme sürecidir. Bu, modelin doğru son yanıtı üretmesine yardımcı olur. *Düşünce ağacı* (***T**ree **o**f **T**houghts*, ToT), CoT'nin daha gelişmiş bir versiyonudur.

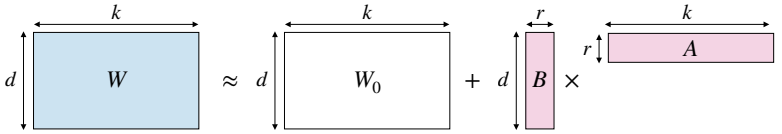
Not: Öz-tutarlılık, CoT akıl yürütme yolları boyunca yanıtları toplayan bir yöntemdir.

3.3 İnce ayarlama

❒ **SFT** – *Denetimli ince ayarlama* (***S**upervised **F**ine**T**uning*, SFT), modelin davranışını bir son görevle hizalayan bir eğitim sonrası yaklaşımıdır. Görevle hizalanmış yüksek kaliteli giriş-çıkış çiftlerine dayanır.

Not: SFT verileri talimatlarla ilgiliyse, bu adıma "talimat uyarlama" denir.

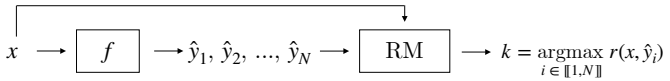
❒ **PEFT** – *Parametre verimli ince ayarlama* (***P**arameter-**E**fficient **F**ine**T**uning*, PEFT), SFT'yi verimli bir şekilde çalıştırmak için kullanılan bir yöntem kategorisidir. Özellikle, *düşük-dereceli adaptasyon* (***Lo**w-**R**ank **A**daptation*, LoRA), W_0'ı sabitleyerek ve bunun yerine düşük sıralı matrisler A, B'yi öğrenerek öğrenilebilir ağırlıklar W'yi yaklaşık olarak hesaplar:



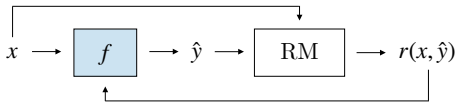
Not: Diğer PEFT teknikleri arasında ön ek ayarlama ve uyarlama katmanı ekleme yer alır.

3.4 Tercih ayarlaması

❒ **Ödül modeli** – *Ödül modeli* (***R**eward **M**odel*, RM), girdi x verildiğinde çıktı $\hat{y}$'nin istenen davranışla ne kadar iyi uyumlu olduğunu tahmin eden bir modeldir. En iyi N örnekleme (Best-of-N sampling, BoN), aynı zamanda reddetme örneklemesi olarak da adlandırılır, N nesil arasında en iyi yanıtı seçmek için bir ödül modeli kullanan bir yöntemdir.



❒ **Pekiştirmeli öğrenme** – *Pekiştirmeli öğrenme* (***R**einforcement **L**earning*, RL), RM'yi kaldıraçlayan ve model f'yi üretilen çıktıları için ödüllere göre güncelleyen bir yaklaşımdır. RM insan tercihlerine dayanıyorsa, bu sürece *insan geri bildirimli pekiştirmeli öğrenme* (***R**einforcement **L**earning from **H**uman **F**eedback*, RLHF) denir.

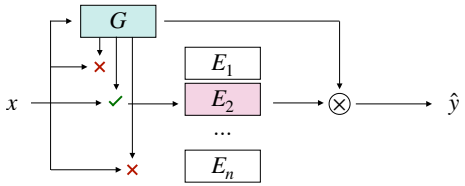


Dengeli politika optimizasyonu (***P**roximal **P**olicy **O**ptimization*, PPO), ödül manipülasyonunu (reward hacking) önlemek için modeli temel modele yakın tutarken daha yüksek ödülleri teşvik eden popüler bir RL algoritmasıdır.

Not: RM ve RL'yi tek bir denetlenen adımda birleştiren doğrudan tercih optimizasyonu (Direct Preference Optimization, DPO) gibi denetlenen yaklaşımlar da vardır.

3.5 Optimizasyonlar

❒ **Uzman karışımları** – *Uzman karışımı* (***M**ixture **o**f **E**xperts*, MoE), çıkarım zamanında nöronlarının yalnızca bir kısmını etkinleştiren bir modeldir. Bir G geçitine ve uzmanlar $E_1, ..., E_n$'e dayanır.



$$\hat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$$

MoE tabanlı BDM'ler FFNN'lerinde bu geçit mekanizmasını kullanırlar.

Not: MoE tabanlı bir BDM'yi eğitmek, çıkarım zamanı verimliliğine rağmen bu mimariyi kullanmamayı seçen yazarların bulunduğu LLaMA makalesinde belirtildiği gibi, oldukça zordur.

❒ **Damıtma** – *Damıtma* (*distillation*), (küçük) bir öğrenci modeli S'nin (büyük) bir öğretmen modeli T'nin tahmin çıktıları üzerinde eğitildiği bir işlemdir. KL ayrışma kaybı kullanılarak eğitilir:

$$\mathrm{KL}(\hat{y}_T||\hat{y}_S) = \sum_i \hat{y}_T^{(i)} \log\left(\frac{\hat{y}_T^{(i)}}{\hat{y}_S^{(i)}}\right)$$

Not: Eğitim etiketleri, sınıf olasılıklarını temsil ettikleri için "esnek" etiketler olarak kabul edilir.

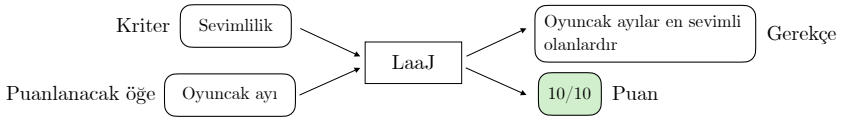
❒ **Kuantalama** – *Model kuantalama*, model ağırlıklarının kesinliğini azaltırken, ortaya çıkan model performansı üzerindeki etkisini sınırlayan bir teknik kategorisidir. Sonuç olarak, bu modelin bellek ayak izini azaltır ve çıkarımını hızlandırır.

Not: QLoRA, LoRA'nın yaygın olarak kullanılan kuantalanmış bir çeşididir.

4 Uygulamalar

4.1 Yargıç olarak BDM

❒ **Tanım** – *Yargıç olarak BDM* (***L**LM-**a**s-**a**-**J**udge*, LaaJ), verilen çıktıları bazı sağlanan kriterlere göre puanlamak için bir LLM kullanan bir yöntemdir. Özellikle, yorumlanabilirliğe yardımcı olan puanı için bir gerekçe de üretebilir.



Özet değerlendirme için duyarlılık-odaklı yardımcısı (***R**ecall-**O**riented **U**nderstudy for **G**isting **E**valuation*, ROUGE) gibi BDM öncesi dönem metriklerinin aksine, LaaJ herhangi bir referans metnine ihtiyaç duymaz, bu da onu her türlü görevde değerlendirmeyi kolaylaştırır. Özellikle, LaaJ iyi performans göstermek için akıl yürütme yetenekleri gerektirdiğinden, büyük ve güçlü bir modele (örn. GPT-4) dayandığında insan derecelendirmeleriyle güçlü bir korelasyon gösterir.

Not: LaaJ, hızlı değerlendirme turları gerçekleştirmek için kullanışlıdır ancak herhangi bir yanlılık olmadığından emin olmak için LaaJ çıktıları ile insan değerlendirmeleri arasındaki uyumu izlemek önemlidir.

❒ **Ortak yanlılıklar** – LaaJ modelleri aşağıdaki yanlılıkları gösterebilir:

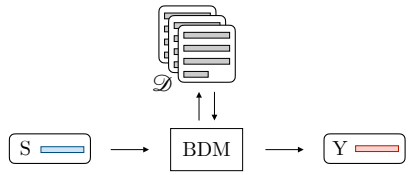
	Konum yanlılığı	**Söz fazlalığı yanlılığı**	**Kendini geliştirme yanlılığı**
Sorun	Çiftler arası karşılaştırmalarda ilk pozisyonu tercih eder	Daha ayrıntılı içerikleri tercih eder	Kendileri tarafından üretilen çıktıları tercih eder
Çözüm	Rastgele pozisyonlarda ortalama metrik	Çıktı uzunluğuna bir ceza eklenir	Farklı bir temel modelden oluşturulmuş bir yargıç kullanılır

Bu sorunlara bir çözüm, özel bir LaaJ'yi ince ayarlama yapmak olabilir, ancak bu çok fazla çaba gerektirir.

Not: Yukarıdaki yanlılıkların listesi kapsamlı değildir.

4.2 RAG

❒ **Tanım** – *Bilgi erişim destekli üretim* (***R**etrieval-**A**ugmented **G**eneration*, RAG), BDM'nin belirli bir soruyu yanıtlamak için ilgili dış bilgiye erişmesine izin veren bir yöntemdir. Bu, özellikle BDM önceden eğitilmiş bilgi kesme tarihinden sonraki bilgileri dahil etmek istediğimizde faydalıdır.



Bir bilgi temelli $\mathscr{D}$ ve bir soru verildiğinde, bir bilgi getirici (retriever) en alakalı belgeleri getirir, ardından çıktıyı üretmeden (generating) önce komutu alakalı bilgilerle zenginleştirir.

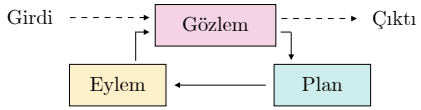
Not: Bilgi erişimi aşaması genellikle yalnızca kodlayıcı modellerden gelen sözcük temsillerine dayanır.

❒ **Hiperparametreler** – Bilgi temelli $\mathscr{D}$, belgelerin n_c boyutundaki parçalara bölünmesi ve bunların $\mathbb{R}^d$ boyutundaki vektörlere sözcük temsilli haline getirilmesiyle başlatılır.

4.3 Temsilci

❒ **Tanım** – Bir *temsilci* (*agent*), hedefleri otonom olarak takip eden ve bir kullanıcı adına görevleri tamamlayan bir sistemdir. Bunu yapmak için farklı BDM sorgu zincirlerini kullanabilir.

❒ **ReAct** – *Mantık yürüt + uygula* (***R**eason + **Act***, ReAct), karmaşık görevleri tamamlamak için birden fazla BDM sorgu zincirine izin veren bir çerçevedir:



Bu çerçeve aşağıdaki adımlardan oluşur:

- *Gözlem*: Önceki eylemleri sentezle ve şu anda bilinenleri açıkça belirtir.
- *Plan*: Hangi görevlerin tamamlanması gerektiğini ve hangi araçların sorgulanacağını ayrıntılı olarak belirtir.
- *Eylem*: Bir API aracılığıyla bir eylem gerçekleştir veya bir bilgi temelli ilgili bilgileri arar.

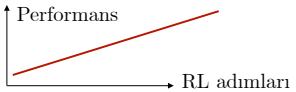
Not: Bir temsilci sistemi değerlendirmek zordur. Ancak bu, hem yerel girdi-çıktılar aracılığıyla bileşen düzeyinde hem de sorgu zincirleri aracılığıyla sistem düzeyinde yapılabilir.

4.4 Akıl yürütme modelleri

❒ **Tanım** – Bir *akıl yürütme modeli* (*reasoning model*), matematik, kodlama ve mantıkta daha karmaşık görevleri çözmek için CoT tabanlı akıl yürütme izlerine dayanan bir modeldir. Akıl yürütme modellerine örnek olarak OpenAI'nin o serisi, DeepSeek-R1 ve Google'ın Gemini Flash Thinking verilebilir.

Not: DeepSeek-R1, akıl yürütme izini `<think>` etiketleri arasında açıkça çıktı olarak verir.

❒ **Ölçekleme** – Akıl yürütme yeteneklerini geliştirmek için iki tür ölçekleme yöntemi kullanılır:

	Tanım	**Gösterim**
Eğitim-süresi ölçeklemesi	Yanıt vermeden önce modelin CoT tarzı akıl yürütme izlerini nasıl üreteceğini öğrenmesini sağlamak için RL'yi daha uzun süre çalıştırın	
Test-süresi ölçeklemesi	Modelin `"Bekle"` gibi bütçe zorlayıcı anahtar sözcüklerle bir yanıt vermeden önce daha uzun süre düşünmesini sağlayın