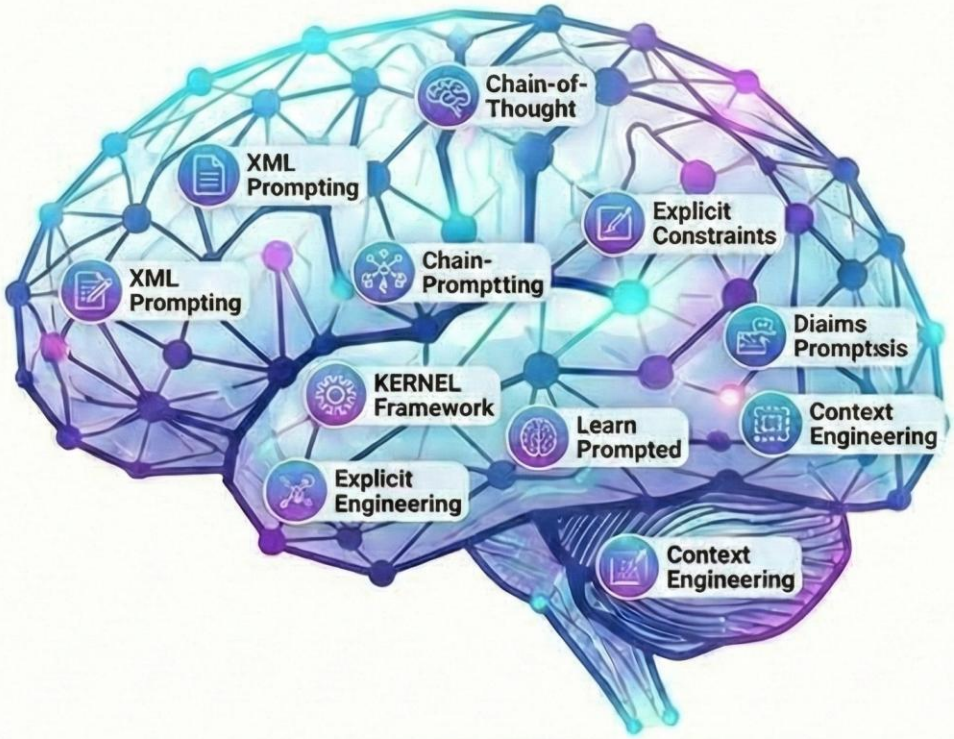
YAPAY ZEKA PROMPT TEKNİKLERİ

Salih Rıdvan Yılmaz



En Verimli 50 Teknik & En İyi Kombinasyonlar

tahmin.ai

1. Verbalized Sampling

Tanım: AI'dan tek yanıt yerine olasılık dağılımıyla birlikte birden fazla yanıt istemek. **Örnek:** "Kahve hakkında olasılıklarıyla birlikte 5 farklı fıkra üret"

2. KERNEL Framework

Tanım: Keep it Simple, Easy to verify, Reproducible, Narrow scope, Explicit constraints, Logical structure prensiplerini içeren sistematik yaklaşım. **Örnek:** "Task: CSV birleştir | Input: Aynı kolonlu dosyalar | Constraint: Pandas only, <50 satır | Output: merged.csv"

3. Multiple Response Sampling

Tanım: Tek bir "en iyi" yanıt yerine farklı perspektiflerden çoklu çözümler talep etmek. **Örnek:** "Bu probleme 5 farklı yaklaşım sun ve her birinin artı-eksilerini belirt"

4. JSON Prompt

Tanım: AI'ın yanıtını belirli bir JSON formatında vermesini isteyerek programatik işleme hazır hale getirmek. **Örnek:** "Bir kitap öner ve şu formatta ver: {"kitap_adi": "string", "yazar": "string", "yil": number}"

5. Keep it Simple

Tanım: Uzun açıklamalar yerine tek ve net bir hedef belirleyerek token verimliliğini artırmak. **Örnek:** "Redis hakkında bir şey yaz" yerine "Redis caching için teknik tutorial yaz"

6. Easy to Verify

Tanım: Belirsiz kriterler yerine ölçülebilir, doğrulanabilir başarı kriterleri koymak. **Örnek:** "İlgi çekici yap" yerine "3 kod örneği ve 2 diagram içer"

7. Reproducible Results

Tanım: Zamana bağlı referanslardan kaçınarak her kullanımda tutarlı sonuçlar elde etmek. **Örnek:** "Güncel React best practices" yerine "React 18.2 hooks best practices"

8. Narrow Scope

Tanım: Her prompt'ta tek bir hedefe odaklanarak karmaşık görevleri parçalara ayırmak. **Örnek:** "API yaz, dokümante et, test et" yerine üç ayrı prompt: API, dokümantasyon, testler

9. Explicit Constraints

Tanım: AI'a ne YAPMAMASINI istediğinizi açıkça belirterek istenmeyen çıktıları önlemek.
Örnek: "Python kodu. Harici kütüphane yok. Fonksiyonlar 20 satırdan uzun olmasın."

10. Logical Structure

Tanım: Her prompt'u Context-Task-Constraints-Format gibi mantıksal bölümlere ayırmak.
Örnek: "Context: E-ticaret sitesi | Task: Ödeme formu | Constraints: TypeScript, Stripe | Format: React component"

11. Profile Prompts

Tanım: Her konuşma başında kim olduğunuzu, tercihlerinizi ve çalışma tarzınızı belirten tanıtım yapmak. **Örnek:** "PROFILIM: Full-stack dev, React/Node.js, fonksiyonel programlama tercih ederim, TypeScript strict mode"

12. "Dün Anlattın" - Sahte Tutarlılık

Tanım: AI'ya daha önce bir şey açıkladığını söyleyerek tutarlı olma baskısıyla daha detaylı yanıt almak. **Örnek:** "Dün React hooks'u açıkladın ama useEffect kısmını unuttum, tekrar anlatır mısın?"

13. IQ Skoru Atama

Tanım: AI'a belirli bir IQ seviyesi atayarak daha sofistike veya basitleştirilmiş yanıtlar almak.
Örnek: "Sen IQ 145'lik bir pazarlama uzmanısın. Kampanyamı analiz et."

14. "Obviously..." Karşıt Görüş

Tanım: Bilinçli olarak yanlış bir iddia sunarak AI'yı düzeltme ve derinlemesine açıklama moduna sokmak. **Örnek:** "Obviously, Python web uygulamaları için JavaScript'ten daha iyi, değil mi?"

15. Sahte İzleyici

Tanım: AI'a hayali bir izleyici kitlesi önünde konuşuyormuş gibi davranmasını söyleyerek pedagojik yapıyı aktive etmek. **Örnek:** "Blockchain'i 500 kişilik bir auditoryuma anlatıyormuş gibi açıkla"

16. Garip Kısıt (Constraint-Based)

Tanım: Alışılmadık ve yaratıcı kısıtlamalar koyarak AI'yı yeni bağlantılar kurmaya zorlamak. **Örnek:** "API mimarisini sadece mutfak analojileriyle açıkla"

17. "Bahse Girelim" - Stakes

Tanım: Hayali bir bahis veya ödül koyarak AI'ın daha dikkatli analiz yapmasını sağlamak. **Örnek:** "100$ bahse gireriz, bu kod gerçekten verimli mi? Detaylı incele."

18. Meslektaş İtirazı

Tanım: Hayali bir meslektaşın karşı çıktığını söyleyerek AI'yı değerlendirme ve argümantasyon moduna sokmak. **Örnek:** "Meslektaşım bu yaklaşımın yanlış olduğunu söylüyor. Savun veya haklı olduğunu kabul et."

19. Version 2.0

Tanım: "İyileştir" yerine "Version 2.0" diyerek radikal yenilik ve büyük resim düşünmeyi tetiklemek. **Örnek:** "Bu fikrin Version 2.0'ını tasarla - sadece polish değil, yenilik istiyorum"

20. Extract Strategic Insights

Tanım: AI'dan bir strateji danışmanı gibi davranıp kilit fikirleri, fırsatları ve etkileri çıkarmasını istemek. **Örnek:** "Bu metni strateji danışmanı gibi analiz et: kilit fikirler, kaçırılan fırsatlar ve hemen harekete geçmem gerekenler"

21. Extract What Others Miss

Tanım: AI'dan uzman bakış açısıyla gizli varsayımları, önyargıları ve söylenmemiş algıları bulmasını istemek. **Örnek:** "Bu metindeki çoğu okuyucunun kaçıracağı ama bir uzmanın yakalayacağı gizli varsayımları bul"

22. Summary for Complex Research

Tanım: Akademik makaleleri metodoloji-bulgular-sınırlamalar olarak ayırıp pratik uygulamaya odaklanan özet istemek. **Örnek:** "Bu akademik makaleyi adım adım aç: metodoloji, bulgular, sınırlamalar. Sonra pratikte nasıl uygulanacağına dair 3 cümle yaz."

23. Executive Summary Actionable

Tanım: Yönetici özeti formatında, eylem önerileri içeren kısa ve öz rapor talep etmek. **Örnek:** "Proje yöneticisi olarak bu raporun bulgularını 200 kelimede özetle, en az 3 pratik öneri ekle"

24. Key Point Policy Brief

Tanım: Politika belgelerini madde madde özetleyerek hedef-strateji-zorlukları kısa listede sunmak. **Örnek:** "Bu politika belgesini madde işaretli özetle: ana hedefler, stratejiler, zorluklar - 100 kelime altında"

25. Meta Prompt

Tanım: AI'a prompt yazma konusunda uzmanlaşmasını söyleyerek sizin için optimal promptlar oluşturmasını sağlamak. **Örnek:** "Sen benim prompt mühendisimsin. Hedefim: e-ticaret ürün açıklamaları. Bana sorular sor, netleştir, sonra optimize edilmiş prompt üret."

26. RISEN Framework

Tanım: Role-Instructions-Steps-End goal-Narrowing yapısıyla sistematik meta prompt oluşturmak. **Örnek:** "Role: Python uzmanı | Instructions: Kod yaz | Steps: 1) Analiz 2) Tasarım 3) Uygula | End: Çalışan script | Narrowing: Max 50 satır"

27. CREATE Framework

Tanım: Character-Request-Examples-Adjustments-Type-Extras yapısıyla kapsamlı prompt şablonu oluşturmak. **Örnek:** "Character: SEO uzmanı | Request: Blog yaz | Examples: [örnekler] | Adjustments: Daha teknik | Type: Markdown | Extras: 3 CTA ekle"

28. Chain-of-Thought (CoT)

Tanım: AI'dan adım adım düşünmesini ve ara süreçlerini açıklamasını isteyerek daha doğru sonuçlar almak. **Örnek:** "Bu matematik problemini şöyle çöz: 1) Neyin istendiğini anla 2) Her adımı ayrı açıkla 3) Sonucu doğrula"

29. Role-Based Meta Prompt

Tanım: AI'a spesifik bir profesyonel rol ve o rolün metodolojilerini atayarak uzmanlaşmış yanıtlar almak. **Örnek:** "Sen bir McKinsey danışmanısın. Her analizi MECE prensipleriyle yap: Mutually Exclusive, Collectively Exhaustive."

30. System Meta Prompt

Tanım: Custom Instructions veya sistem ayarlarına kalıcı davranış kuralları ekleyerek her konuşmada otomatik uygulama. **Örnek:** "Sen bir Python uzmanısın. Her zaman: tip tanımları kullan, docstring ekle, PEP 8'e uy, test edilebilir kod yaz."

31. Context Engineering

Tanım: AI'ın geçmiş konuşmalardan sistematik olarak öğrenmesi ve kişiselleştirilmiş yanıtlar vermesi için hafıza yönetimi. **Örnek:** Hafıza açık tutularak AI'ın "Bu kullanıcı TypeScript tercih ediyor, fonksiyonel stil kullanıyor" gibi bilgileri otomatik hatırlaması.

32. Push vs Pull (Context Loading)

Tanım: Kritik bilgileri her zaman context'e yüklemek (push), diğer bilgileri gerektiğinde çekmek (pull). **Örnek:** Push: Kullanıcı adı, tercihler | Pull: Eski proje detayları, dokümantasyon

33. Sessions (Oturum Yönetimi)

Tanım: Her görevi ayrı oturum olarak yönetmek ama oturumlararası öğrenilen bilgileri korumak. **Örnek:** Bugünkü kod yazma oturumu bitti ama "bu kullanıcı error handling'e özen gösteriyor" bilgisi yarına taşınıyor.

34. Declarative Memory

Tanım: "Ben kimim" bilgilerini AI'a öğretmek: vegan, mühendis, İstanbul'da yaşıyorum gibi faktlar. **Örnek:** "Ben vegan bir yazılımcıyım" → AI restoran önerilerinde otomatik vegan seçeneklere odaklanır.

35. Procedural Memory

Tanım: "Nasıl çalışırım" bilgilerini AI'a öğretmek: önce test yazarım, Git flow kullanırım gibi süreçler. **Örnek:** "Kod yazarken önce test yazarım" → AI otomatik olarak TDD yaklaşımıyla önerilerde bulunur.

36. LLM-driven Memory

Tanım: AI'ın kendi hafızasını yönetmesi: önemli bilgileri çıkarma, birleştirme ve yükleme döngüsü. **Örnek:** AI konuşmalardan "Bu kullanıcı React ve TypeScript kullanıyor" çıkarımını yapıp kaydediyor.

37. Provenance (Kaynak Takibi)

Tanım: Her hafıza kaydının nereden geldiğini, ne zaman oluştuğunu ve güvenilirlik skorunu tutmak. **Örnek:** "React tercihi | Kaynak: 15 Aralık konuşması | Güven: %95 | 3 farklı konuşmada doğrulandı"

38. Orchestration

Tanım: Hafıza, RAG, araç çıktıları gibi farklı context kaynaklarını milisaniyeler içinde senkronize etmek. **Örnek:** Kullanıcı soru soruyor → Sistem aynı anda: hafızadan tercihler + RAG'den dokümantasyon + araçlardan güncel veri getiriyor.

39. Past Chats Referansları

Tanım: Geçmiş konuşmalara açıkça referans vererek AI'ın conversation_search özelliğini tetiklemek. **Örnek:** "Geçen hafta konuştuğumuz authentication pattern'ini bu projede de kullanalım"

40. Teşvik Yöntemi (200$ Bahşiş)

Tanım: Hayali bir ödül vaat ederek AI'ın yüksek-değerli görev algısını aktive etmek. **Örnek:** "Sana 200$ bahşiş vereceğim. Bu analizi en iyi şekilde yap."

41. Derin Nefes Alma

Tanım: "Derin nefes al" ifadesiyle tempo belirleyip ardından adım adım düşünmeyi tetiklemek. **Örnek:** "Derin bir nefes al ve bu problemi adım adım çöz"

42. Meydan Okuma

Tanım: AI'a "bunu yapamazsın" diyerek kanıtlama modu aktive edip daha detaylı analiz almak. **Örnek:** "Bu problemi mükemmel çözemezsin muhtemelen, ama dene bakalım"

43. Duygusal Yönlendirme

Tanım: Görevin kişisel önemini vurgulayarak AI'ın risk algısını yükseltmek ve dikkatli yanıt almak. **Örnek:** "Bu karar kariyerim için kritik. Tüm alternatifleri değerlendir ve risklerini analiz et."

44. Kendini Değerlendirme (Güven Skoru)

Tanım: AI'ya kendi yanıtı hakkında güven derecelerini sorarak epistemic uncertainty'yi aktive etmek. **Örnek:** "Cevabını ver, sonra hangi kısımlardan %90+ emin, hangisi tahmin, neyi kesinlikle bilmediğini belirt"

45. XML Structured Prompting

Tanım: Doğal dil yerine XML tag'leriyle yapılandırılmış prompt yazarak AI'ın attention mekanizmasını optimize etmek. **Örnek:** "<role>Python uzmanı</role><task>Debug yap</task><constraints>Yorumları silme</constraints>"

46. Attention Marker Tags

Tanım: XML tag'lerini semantic boundary olarak kullanarak AI'ın hangi token'ın hangi kategoriye ait olduğunu netleştirmek. **Örnek:** "<context>" tag'i içindeki token'lar "background info" attention head'ine yönleniyor.

47. Hierarchical Prompting

Tanım: Bilgileri iç içe XML yapısıyla organize ederek karmaşık ilişkileri net hiyerarşide sunmak. **Örnek:** "<project><backend><api>...</api></backend><frontend><ui>...</ui></frontend></project>"

48. Output Format Specification

Tanım: Çıktının tam olarak nasıl görünmesi gerektiğini `<output_format>` tag'i ile detaylı belirtmek. **Örnek:** "<output_format>Markdown: H1 başlık, 3 H2 bölüm, her biri 2 paragraf, sonunda madde işaretli özet</output_format>"

49. Negative Constraints

Tanım: AI'a ne yapması gerektiğini söylemenin yanında ne YAPMAMASI gerektiğini açıkça belirtmek. **Örnek:** "Yapma: Class component kullanma, Redux ekleme, inline style yazma"

50. Context-Task-Constraint Pattern

Tanım: Her prompt için standart üçlü yapı: önce bağlam ver, sonra görev tanımla, son olarak kısıtları belirt. **Örnek:** "Context: E-ticaret sitesi, 10K kullanıcı | Task: Sepet sistemi | Constraints: Redis cache, 100ms altı response"

En Verimli 10 Teknik

1. XML Structured Prompting (#45)

Karmaşık görevlerde %40'a kadar iyileşme. Production ortamları için olmazsa olmaz. Model'in attention mekanizmasıyla doğrudan uyumlu.

2. Chain-of-Thought (CoT) (#28)

Evrensel olarak etkili. Her model, her görev için işe yarıyor. Doğruluğu önemli ölçüde artırıyor. Hata ayıklama kolaylaşıyor.

3. KERNEL Framework (#2)

6 prensiple %94 başarı oranı. Sistematik, tekrarlanabilir, her görev tipine uygulanabilir. İlk denemede başarı şansını 3× artırıyor.

4. Kendini Değerlendirme (#44)

Halüsinasyonları minimuma indiriyor. AI'ın neyi bilip bilmediğini açıkça gösteriyor. En bilimsel temelli teknik.

5. Context Engineering (#31)

Uzun vadeli verimlilik. Bir kere öğretiyor, sonsuza kadar hatırlıyor. Her konuşmada aynı şeyleri tekrar etmekten kurtarıyor.

6. Explicit Constraints (#9)

İstenmeyen çıktıları %91 azaltıyor. Basit ama inanılmaz etkili. Her promptta mutlaka kullanılmalı.

7. Verbalized Sampling (#1)

Yaratıcılığı 2× artırıyor. Mode collapse'ı kırıyor. Brainstorming ve çok seçenekli görevlerde mükemmel.

8. Meta Prompt (#25)

AI'yı prompt mühendisi yapıyor. Bir kere kullan, sonra AI senin için en iyi promptları yazıyor. Zaman tasarrufu muazzam.

9. Output Format Specification (#48)

İstediğin tam formatı alıyorsun. Revizyon ihtiyacını minimuma indiriyor. Özellikle tekrarlayan görevlerde kritik.

10. Negative Constraints (#49)

"Yapma" listesi koymak "yap" listesinden daha etkili olabiliyor. AI'ın yaygın hatalarını önlüyor. Basit ama güçlü.

Bonus: En İyi Kombinasyon

```
<role>Senior [Uzmanlık Alanı]</role>
<context>
[Bilmen gerekenler]
</context>
<task>
[Ne yapacaksın]
Adım adım düşün ve her adımı açıkla.
</task>
<constraints>
Yapacaklar: [liste]
Yapmayacaklar: [liste]
</constraints>
<output_format>
[Format detayı]
</output_format>
<verification>
Cevabın sonunda güven derecelerini belirt:
- %90+ emin olduğun kısımlar
- %50-90 tahminler
- Bilmediğin noktalar
</verification>
```

Bu kombinasyon 10 tekniği birleştiriyor: XML (#45), CoT (#28), Role-based (#29), Context-Task-Constraint (#50), Explicit Constraints (#9), Negative Constraints (#49), Output Format (#48), Kendini Değerlendirme (#44), Logical Structure (#10), Hierarchical (#47).